Oracle DBA Made Simple

Oracle database administration techniques

Mike Ault

Retail Price $19.95 US/$29.95 Canada
ISBN: 0-9740716-5-X
Copyright © 2003 by Rampant TechPress

Rampant TechPress

Oracle DBA made simple

Oracle database administration techniques

Mike Ault

Notice

While the author makes every effort to ensure the information presented in this white paper is accurate and without error, Rampant TechPress, its authors and its affiliates takes no responsibility for the use of the information, tips, techniques or technologies contained in this white paper. The user of this white paper is solely responsible for the consequences of the utilization of the information, tips, techniques or technologies reported herein.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Oracle DBA made simple

Oracle database administration techniques

By Mike Ault

Copyright © 2003 by Rampant TechPress. All rights reserved.

Published by Rampant TechPress, Kittrell, North Carolina, USA

Series Editor: Don Burleson

Production Editor: Teri Wade

Cover Design: Bryan Hoff

Oracle, Oracle7, Oracle8, Oracle8i, and Oracle9i are trademarks of Oracle Corporation. *Oracle In-Focus* is a registered Trademark of Rampant TechPress.

Many of the designations used by computer vendors to distinguish their products are claimed as Trademarks. All names known to Rampant TechPress to be trademark names appear in this text as initial caps.

The information provided by the authors of this work is believed to be accurate and reliable, but because of the possibility of human error by our authors and staff, Rampant TechPress cannot guarantee the accuracy or completeness of any information included in this work and is not responsible for any errors, omissions, or inaccurate results obtained from the use of information or scripts in this work.

Visit **www.rampant.cc** for information on other *Oracle In-Focus* books.

ISBN: 0-9740716-5-X

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Table Of Contents

Notice ii

Publication Information iii

Table Of Contents iv

Introduction 1

System Planning 1

Resource and Capacity Planning 2
Resource Specification for Oracle 2
Optimal Flexible Architecture (OFA) 7
Minimum OFA Configuration 8
Oracle Structures and How They Affect Installation 10
Executables 11
Data Files 11
Redo Logs 12
Control Files 12
Exports 13
Archive Logs 13
LOB Storage 14
BFILE Storage 14
Disk Striping, Shadowing, RAID, and Other Topics 14
Disk Striping 15
Disk Shadowing or Mirroring 16
RAID—Redundant Arrays of Inexpensive Disks 17
New Technologies 18
Optical Disk Systems 18
Tape Systems 19
RAM Drives (Random Access Memory) 19

Backup & recovery 20

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Backups 21
Cold Backups 21
Hot Backups 23
Example Documentation Procedure for NT Online Backup and Recovery Scripts 48
Imports/Exports 51
Limitations on export/import: 51
Exports 51
IMPORT 53
Archive Logs 54
Backup Methodologies 56
NT or UNIX System Backup 56
Import/Export 57
Archive Logging 57
Recovery Types 57
Oracle7 Enterprise Backup Utility 59
Oracle8 RECOVERY MANAGER FACILITY 63
Installing the RMAN Catalog 66
Incomplete restore scenario 69
DB_VERIFY UTILITY 71
The following example shows how to get on-line help: 72
The DBMS_REPAIR Utility 73
DBMS_REPAIR Enumeration Types 74
DBMS_REPAIR Exceptions 74
DBMS_REPAIR Procedures 75
ADMIN_TABLES 76
CHECK_OBJECT 81
DUMP_ORPHAN_KEYS 84
FIX_CORRUPT_BLOCKS 85
REBUILD_FREELISTS 86
SKIP_CORRUPT_BLOCKS 88
Oracle RDBMS Architecture 89
Background Processes 90

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Datafiles 94
Datafile Sizing 96
Rollback Segments 102
Redo log files 104
Control files 105
Initialization File 105
The Undocumented Initialization Parameters ("_") 126
The Initialization File Event Settings 145
System Global Area 157
SGA 157
Modifying the INIT.ORA 158
Allocating And Caching Memory 159
Use of the Default Pool 159
Use of The KEEP Pool 160
Use of the RECYCLE Pool 160
Tuning the Three Pools 161
Shared Pool 162
Putting it All In Perspective 180
What to Pin 185
The Shared Pool and MTS 190
Large Pool Sizing 191
A Matter Of Hashing 194
Disk IO and the Shared Pool 198
Monitoring Library and Data Dictionary Caches 201
In Summary 204
Managing the Database 209
Find USER locking others/Kill problem USER 209
Methods of Murder 213
Killing From the Oracle Side 213
Killing From the Operating System Side 215
Creating and starting the database 215
Database Creation 216
Re-creation of a Database 219
Database Startup and Shutdown 229

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Startup 230
Shutdown 232
Tuning Responsibilities 234
Step 1: Tune the Business Rules 234
Step 2: Tune the Data Design 234
Step 3: Tune the Application Design 235
Step 4: Tune the Logical Structure of the Database 235
Step 5: Tune Database Operations 235
Step 6: Tune the Access Paths 236
Step 7: Tune Memory Allocation 236
Step 8: Tune I/O and Physical Structure 237
Step 9: Tune Resource Contention 238
Step 10: Tune the Underlying Platform(s) 238
Tuning Summary 238
Layout & Fragmentation 239
Tablespace Segments & Free Space 241
Tables & Indexes/Partitioning 242
The V$ views 243
How are they used? 244
The Optimizers & the Analyze Command 248
RULE Based Optimizer 249
COST Based Optimizer 250
The Parallel Query Option 253
Parallel query settings 253
Problems In Parallel Query Usage 258
Security 258
Users 258
Altering Users 264
Dropping Users 265
Grants 266
System Privileges 267
Object Privileges 276
Column Privileges 281
View Grants 283

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Other Grants 283
Revoking Grants 283
Use Of Roles 285
Creating Roles 286
Grants To Roles 287
Setting Roles 289
Special Roles 290
OSOPER And OSDBA 292
CONNECT, RESOURCE, And DBA Roles 293
Export/Import Roles 294
Using PROFILES 295
Profiles and Resource Limits 297
Altering Profiles 300
Profiles and Passwords 301
Managing CPU Utilization for in Oracle8i 303
Creating a Resource Plan 304
Restricting Access by Rows in Oracle8i 329
Policy Usage 335
DBMS_RLS Package 337
Summary 340

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Introduction

Database Administration has grown over the years from the mere management of a few tables and indexes to a complex interlocking set of responsibilities ranging from managing database objects to participating in enterprise wide decisions on hardware, software and development tools.

In order to fully perform these functions the modern Oracle DBA needs a large skill set. Over the next few hours we will discuss the skills needed and specifically how they apply to an Oracle DBA.

System Planning

In a green field operation (one where you are there before the equipment and database) a DBA will have a critical part to play in the planning and configuration of the new system. Even in existing environments the ability to plan for new servers, new databases or improvements to existing databases and systems is critical.

Essentially the DBA must concern themselves with two major issues:

1. Get enough server to insure adequate performance
2. Allow for enough backup and recovery horsepower to get backup and recovery performed within the required time constraints.

All of this actually falls under the topic resource and capacity planning.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Resource and Capacity Planning

Oracle is a resource intensive database system. The more memory, CPU and disk resources you can provide Oracle, the better it performs. Resource planning with Oracle becomes more a game of "how much can we afford to buy" instead of "what is the minimum configuration". A minimally configured Oracle server will not function in an efficient manner.

Resource Specification for Oracle

In resource specification there are several questions that must be answered.

1. How many users will be using the system both now and in the future?
2. How much data will the system contain both now and in the future, do we know growth rates?
3. What response times are expected?
4. What system availability is expected?

Why are these questions important?

1. How many users will be using the system both now and in the future?

   This question is important because it effects how much processing power is going to be required. The number of users will determine number and speed of CPUs, size of memory, network related configuration.

2. How much data will the system contain both now and in the future, do we know growth rates?

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

This question is important because it determines disk needs, how much storage will be required to take data we have today and how much will be needed to allow for growth. The answer to this question also helps determine how much memory will be required.

3. What response times are expected?

   This question is important because it drives number, type and speed of CPU resources as well as network issues. In addition it will drive disk configuration issues such as number and speed of disks, number and speed of controllers, disk partitioning decisions.

4. What system availability is expected?

   This question is important because system availability drives the type of RAID configuration (1, 0, 0/1, RAID5), the type of backup expected (cold, hot) and any parallel server issues. The requirements change if all that is expected is the system be available during working hours Monday through Friday or if the system is expected to be available 24X7 seven days a week. This also drives the type of backup media, whether a single tape drive is all that is required or do we need a hi-speed, multi-channel, tape-stacker, silo based solution?

To properly perform capacity planning a cooperative effort must be undertaken between the system administrators, database administrators and network administrators.

Step 1: Size the Oracle database

A starting point for the whole process of capacity planning is to know how many and what size databases will be supported on a given server resource. The physical sizing of tables, indexes,

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

clusters and LOB storage areas will play a critical role in sizing the overall database including the shared global memory areas and disk farm. The DBA and designers must work in concert to accurately size the database physical files. The design of the database will also drive the placement and number of tablespaces and other database resources such as size and quantity of redo logs, rollback segments and their associated buffer areas.

Generally the database block buffer areas of a database SGA will size out at between 1/20 to 1/100 the physical sum of the total number of database file sizes. For example if the database physical size is 20 gigabytes the database block buffers should size out to around 200 megabytes to 1 gigabyte in size depending on how the data is being used. In most cases the SGA shared pool would size out at around 20-150 megabytes maximum depending on the usage model for the shared SQL areas (covered in a later lesson.) For a 20 gigabyte system the redo logs would most likely run between 20 and 80 megabytes, you would want them mirrored and probably no fewer than 5 groups. The log buffer to support a 50 megabyte redo log file would be a minimum of 5 megabytes maybe as large as 10 megabytes. The final major factor for the SGA would be the size of the sort area, for this size of a database a 10-20 megabyte sort area is about right (depending on the number and size of sorts). Remember that sort areas can either be a part of the shared pool or a part of the large pool, this too we will cover in a later lesson.

So based on the above what have we determined? Lets choose 400 megabytes for our database block buffer size, 70 megabytes for the shared pool, 4-10 megabyte log buffers (40 megabytes) and a sort area size of 10 megabytes. We are looking at a 500-600 megabyte SGA with the other non-DBA sizable factors added in. Since you are not supposed to use more than 60% of physical memory (depending on who you ask) this means w will need at least a gigabyte of RAM. With this size of database a single CPU probably won't give sufficient performance so we are probably

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

looking for at least a 4-processor machine. If we have more than one instance installed on the server, the memory requirements will go up.

Step 2: Determine Number and Type of Users:

Naturally a one user database will require fewer resources than a thousand user database. Generally you will need to take a SWAG at how much memory and disk resources each user will require. An example would be to assume that of an installed user base of 1000 users, only 10 percent of them will be concurrently using the database. This leaves 100 concurrent users, of those maybe a second 10 percent will be doing activities that require sort areas, this brings the number down to 10 users each using (from our previous example) 10 megabytes of memory each (100 megabytes.) In addition each of the 100 concurrent users needs approximately 200k of process space (depending on activity, OS and other factors) so we are talking an additional load of 20 megabytes just for user process space. Finally, each of these users will probably require some amount of disk resource (less if they are client-server or web based) let's give them 5 meg of disk to start apiece, that adds up to 5 gigabytes of disk (give or take a meg or two.)

Step 3: Determine Hardware Requirements to Meet Required Response Times and Support User Load:

This step will involve the system administrator and perhaps the hardware vendor. Given our 1000:100:10 mix of users and any required response times numbers they should be able to configure a server that will provide proper performance. Usually this will require multiple, multiple-channel disk interfaces and several physically separate disk arrays.

Step 4: Determine Backup Hardware to Support Required Uptime Requirements:

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Here again the system administrator and hardware vendor will have a hand in the decision. Based on the size of disks and the speed of the backup solution maximum recovery time should be developed. If there is no way to meet required uptime requirements using simple backup schemes then more esoteric architectures may be indicated such as multi-channel tapes, hot standby databases or even Oracle Parallel Server. Let's say we require a 24X7 uptime requirement with instantaneous failover (no recovery time due to the mission critical nature of the system.) This type of specification would require Oracle Parallel Server in an automated failover setup. We would also use either a double or triple disk mirror so that we could split the mirror to perform backups without losing the protection of the mirroring.

Let's compile what we have determined so far:

Hardware: 2 - 4 CPU (at highest speed CPU we can afford) with at least 1 gigabyte (preferably 2) of shared RAM, at least 2 disk controllers each with multiple channels, 90 gigabytes of disk resource using a three way mirror to give us one 30 gig triple mirrored array. The systems themselves should have an internal disk subsystem sufficient to support the operating system and any swap and paging requirements. Systems must be able to share disk resources so must support clustering. High-speed tape backup to minimize mirror-split times.

Software: Oracle Parallel Server, Cluster management software, Networking software, Backup software to support backup hardware.

Capacity and resource planning is not an exact science. Essentially we are shooting for a moving target. The dual Pentium II 200 NT server with 10 gig of 2-gigabyte SCSI disks I bought 2 years ago for $5k has a modern equivalent in the Pentium III 400 with internal 14 gig drive my father-in-law just purchased for $1k. By the time we specify and purchase a system it is already superceded. You

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

should insist on being allowed to substitute more efficient, lower cost options as they come available during the specification and procurement phases.

Optimal Flexible Architecture (OFA)

Optimal Flexible Architecture provides a logical physical layout for the database that helps the DBA to manage the system. In addition, a properly configured Oracle instance will minimize contention thus improving performance. Perhaps one of the most overlooked tuning option, configuration, must utilize OFA guidelines to be successful.

In accordance with Cary V. Millsap of the Oracle National Technical Response Team, the OFA process involves following 3 rules:

1. Establish an orderly operating system directory structure in which any database file can be stored on any disk resource.

    a. Name all devices that might contain Oracle data in such a manner that a wild card or similar mechanism can be used to refer to the collection of devices as a unit.
    b. Make a directory explicitly for storage of Oracle data at the same level on each of these devices.
    c. Beneath the Oracle data directory on each device, make a directory for each different Oracle database on the system.
    d. Put a file *X* in the directory /u??/ORACLE/D (or on VMS DISK2:[ORACLE.D]) if and only if *X* is a control file, redo log file, or data file of the Oracle Database whose DB_NAME is D. *X* is any database file.

    Note: You may wish to add an additional directory layer if you will have multiple Oracle versions running at the same time. This additional layer includes the version level.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

2. Separate groups of segments (data objects) with different behavior into different tablespaces.
   a. Separate groups of objects with different fragmentation characteristics in different tablespaces. (e.g., don't put data and rollback segments together.)
   b. Separate groups of segments that will contend for disk resources in different tablespaces. (e.g., don't put data and indexes together.)
   c. Separate groups of segments representing objects with differing behavioral characteristics in different tablespaces. (e.g., Don't put tables that require daily backup in the same tablespace with ones that require yearly backup.)
   d. Maximize database reliability and performance by separating database components across different disk resources. A caveat for RAID environments, consider also spread across controller volume groups.
      i. Keep at least three active copies of a database control file on at least three different physical drives.
      ii. Use at least three groups of redo logs in ORACLE7. Isolate them to the greatest extent possible on hardware serving few or no files that will be active while the RDBMS is in use. Shadow redo logs whenever possible.
      iii. Separate tablespaces whose data will participate in disk resource contention across different physical disk resources. (You should also consider disk controller usage.)

Minimum OFA Configuration

The minimum suggested configuration would consist of seven data areas, either disks, striped sets, RAID sets, or whatever else comes down the pike in the next few years. These areas should be as

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

separate as possible, ideally operating off of different device controllers or channels to maximize throughput. The more heads you have moving at one time, the faster your database will be. The disk layout should minimize disk contention. For example:

- AREA1: Oracle executables and user areas, a control file, the SYSTEM tablespace, redo logs
- AREA2: Data-data files, a control file, tool-data files, redo logs
- AREA3: Index-data files, a control file, redo logs
- AREA4: Rollback segment-data files
- AREA5: Archive log files
- AREA6: Export Files
- AREA7: Backup Staging

Of course, this is just a start, you mind find it wise to add more areas to further isolate large or active tables into their own areas as well as separating active index areas from each other. Note that on a modern system this configuration may require 4-2 channel controller cards and 8 physically separable disk arrays.
The structure on UNIX could look like the following:

```
/oracle0/product/oracle/8.1.3/            Top level
$ORACLE_HOME
                                              bin/
Standard distribution structure under version
                                              doc/
                                               rdbms/
                                               …
/oracle0/data/                         Place instance names
under type directories
                     ortest1/
                     ortest2/
/oracle0/control/
                     ortest1/
                      ortest2/
/oracle0/redo/
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
                    ortest1/
                    ortest1/
/oracle0/admin/
                      ortest1/
                                bdump/
      backup_dump_dest
                                udump/
user_dump_dest
                                cdump/
core_dump_dest
                                pfile/
      initialization file location (linked back to dbs
directory)
                                create/
      Database creation script storage area
                     ortest2/
                             …
/oracle1/data/
            /control/
            /redo/
/oracle2/data/
            /control/
            /redo/
…
/oracle7/data/
            /control/
            /redo/
```

Using this type of structure even on a RAID5 volume allows for a logical separation of files for ease in locating and controlling database files. For other platforms just alter the directory syntax for example on NT the "/oracle0/product/oracle/8.1.3" directory becomes "c:\oracle0\product\oracle\813\".

Oracle Structures and How They Affect Installation

As can be seen from the previous section, an Oracle database is not a simple construct. Much thought must go into file placement, size, number of control files, and numerous other structural issues before installation. It is a testament to the resiliency of the Oracle

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

RDBMS that even if most of the decisions are made incorrectly, the database that results will still function, albeit, inefficiently.

The structures are as follows:

Oracle executables
Data files—data, index, temporary, rollback
Redo logs
Control files
Export files
Archive logs
Placement of any LOB or BFILE storage structures

Let's examine each of these.

Executables

The Oracle executables are the heart of the system. Without the executables the system is of course worthless since the data files are only readable by Oracle processes. The Oracle executables should be on a disk reserved for executables and maybe some user files. Disk speed is not a big issue, but availability is of major concern. The executables will require 150 to over 200 megabytes or more of disk space. The installation process will create a directory structure starting at a user-specified root directory. There will usually be a subdirectory for each major product installed.

Data Files

Data files are the physical implementations of Oracle tablespaces. Tablespaces are the logical units of storage that would roughly compare to volume groups in an operating system. Each tablespace can have hundreds of tables, indexes, rollback segments, constraints, and other internal structures mapped into it. In return, these are then mapped into the data files that correspond to the tablespaces. Only a limited number of data files can be associated with a tablespace. The total number of data files for the

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

entire database is set by the MAXDATAFILES parameter at creation (VMS defaults to 32, UNIX, 16, NT 32).

Redo Logs

As their name implies, redo logs are used to restore transactions after a system crash or other system failure. The redo logs store data about transactions that alter database information. According to Oracle each database should have at least two groups of two logs each on separate physical non-RAID5 drives; if no archive logging is taking place, three or more groups with archive logging in effect. These are relatively active files and if made unavailable, the database cannot function. They can be placed anywhere except in the same location as the archive logs. Archive logs are archive copies of filled redo logs and are used for point-in-time recovery from a major disk or system failure. Since they are backups of the redo logs it would not be logical to place the redo logs and archives in the same physical location. Size of the redo logs will determine how much data is lost for a disaster affecting the database. I have found three sets of multiplexed logs to be the absolute minimum to prevent checkpoint problems and other redo related wait conditions, under archive log use three groups is a requirement.

Control Files

An Oracle database cannot be started without at least one control file. The control file contains data on system structures, log status, transaction numbers and other important information about the database. The control file is generally less than one megabyte in size. It is wise to have at least two copies of your control file on different disks, three for OFA compliance. Oracle will maintain them as mirror images of each other. This ensures that loss of a single control file will not knock your database out of the water. You cannot bring a control file back from a backup; it is a living file that corresponds to current database status. In both Oracle7 and Oracle8, there is a CREATE CONTROL FILE command that allows recovery from loss of a control file. However, you must have

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

detailed knowledge of your database to use it properly. The section of the recovery chapter that deals with backup and recovery of control files explains in detail how to protect yourself from loss of a control file. It is easier to maintain extra control file copies. In Oracle8 and Oracle8i the use of RMAN may drive control file sizes to tens of megbytes.

Exports

Export files affect the recoverability of your database should some disaster befall it. Export files, created by the export utility supplied by Oracle, are copies of a database's data and structure at a given point in time. There are several types of exports that will be covered in the section on backup and recovery. Export files should be stored in a separate location from archive files.

Archive Logs

Archive logs, as was stated before, are archived copies of the redo logs. They provide the capability to recover to a specific point-in-time for any tablespace in the database. For any application considered to be production or mission-critical, archive logging must be turned on. These files can be stored to disk, tape, or even optical storage such as WORM. Using operating system backups such as BACKUP on VMS or TAR on UNIX, and with the application of archive logs, a database can be quickly recovered after disaster. Archive logs can only be used to recover when cold or hot backups are used for Oracle backup.

After each successful hot or cold backup of an Oracle database, the associated archive and export files may be removed and either placed in storage or deleted. In an active database these files may average tens of megabytes per day; storage for this amount of data needs to be planned for. Just for example, at one installation doing Oracle development with no active production databases, 244 megabytes of archives and over 170 megabytes of exports were generated in a one-week period. If archive logging is turned on, and

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

you run out of archive disk space, the database stops after the last redo log is filled. Plan ahead and monitor disk usage for instances using archive logging. These days gigabytes of logs per day are a normal occurance, besure to provide for this.

LOB Storage

Actually a special form of tablespace, the LOB storage should be on fast disk resources simply due the large required sizes of most LOB (Large Object) data items. LOB storage should be placed away from other types of storage and should only contain LOB and LOB indexe (LOB, CLOB, NCLOB and BLOB) data items.

BFILE Storage

A BFILE is a pointer to an external LOB file. Generally the same considerations given to LOB storage will apply to the storage areas that BFILEs point towards.

Disk Striping, Shadowing, RAID, and Other Topics

Unless you've been living in seclusion from the computer mainstream, you will have heard of the above topics. Let's take a brief look at them and how they will affect Oracle tuning.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

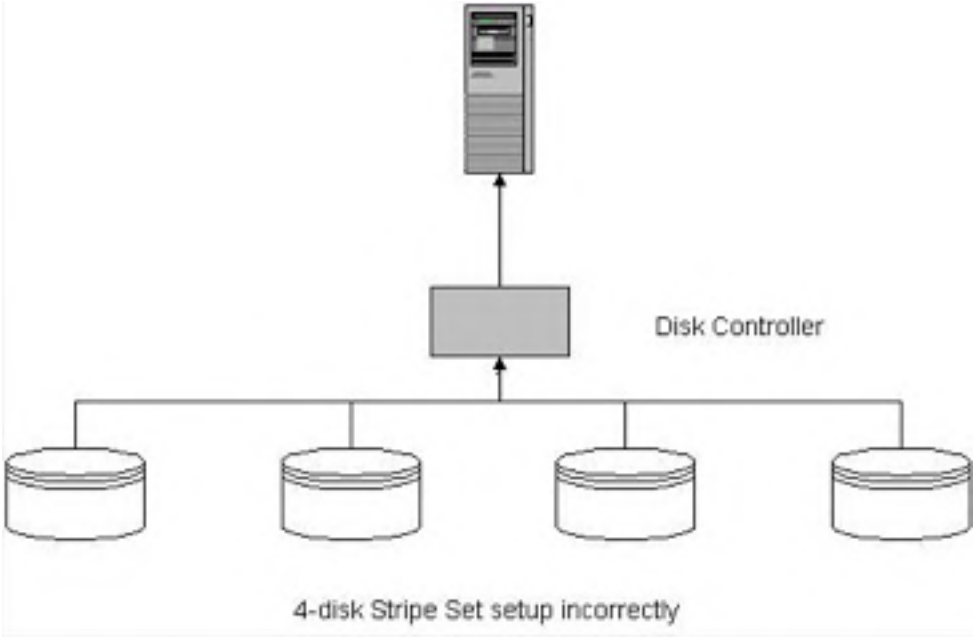


Figure 1: Example Of Improper Striping

Disk Striping

Disk striping is the process by which multiple smaller disks are made to look like one large disk. This allows extremely large databases, or even extremely large single-table tablespaces, to occupy one logical device. This makes managing the resource easier since backups only have to address one logical volume instead of several. This also provides the advantage of spreading IO across several disks. If you will need several gigabytes of disk storage for your application, striping may be the way to go. One disadvantage to striping: If one of the disks in the set crashes, you lose them all unless you have a high reliability array with hot swap capability.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

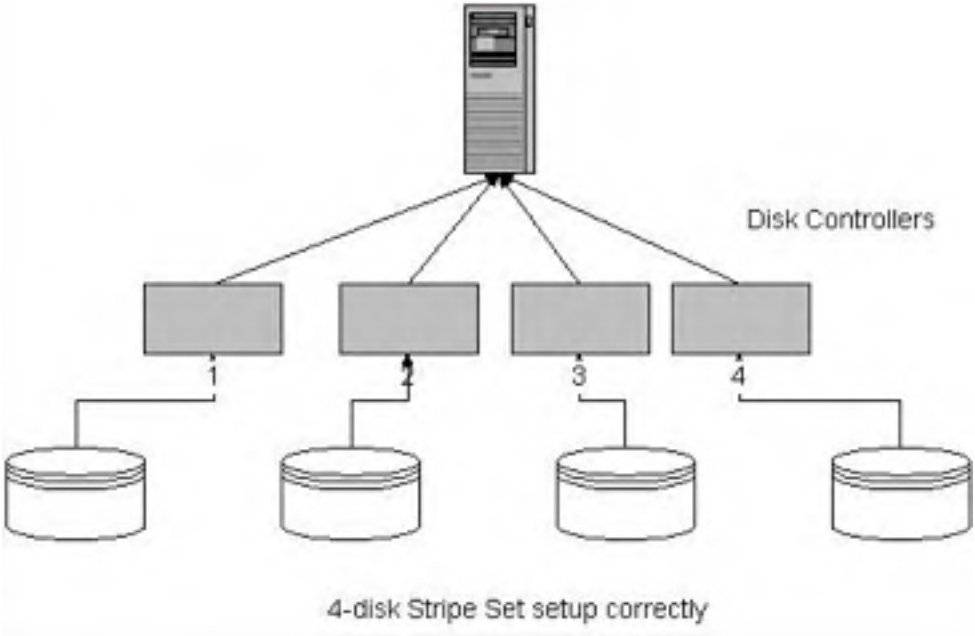


Figure 2: Example of Proper Striping

Disk Shadowing or Mirroring

If you will have mission-critical applications that you absolutely cannot allow to go down, consider disk shadowing or mirroring. As its name implies, disk shadowing or mirroring is the process whereby each disk has a shadow or mirror disk that data is written to simultaneously. This redundant storage allows the shadow disk or set of disks to pick up the load in case of a disk crash on the primary disk or disks; thus the users never see a crashed disk. Once the disk is brought back on-line, the shadow or mirror process brings it back in sync by a process appropriately called “resilvering.” This also allows for backup, since the shadow or mirror set can be broken (e.g., the shadow separated from the primary), a backup taken, and then the set resynchronized. I have heard of two, three and even higher mirror sets, generally I see no reason for more than a three-way mirror as this allows for the set of three to be broken into a single and a double set for backup purposes.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The main disadvantage to disk shadowing is the cost: For a two hundred-gigabyte disk farm, you need to purchase four hundred or more gigabytes of disk storage.

RAID—Redundant Arrays of Inexpensive Disks

The main strength of RAID technology is its dependability. In a RAID 5 array, the data is stored as are check sums and other information about the contents of each disk in the array. If one disk is lost, the others can use this stored information to recreate the lost data. This makes RAID very attractive. RAID has the same advantages as shadowing and striping at a lower cost. It has been suggested that if the manufacturers would use slightly more expensive disks (RASMED—redundant array of slightly more expensive disks) performance gains could be realized. A RAID system appears as one very large, reliable disk to the CPU. There are several levels of RAID to date:

- RAID-0—Known as disk striping
- RAID-1—Known as disk shadowing
- RAID-0/1—Combination of RAID-0 and RAID-1
- RAID-2—Data is distributed in extremely small increments across all disks and adds one or more disks that contain a Hamming code for redundancy. RAID-2 is not considered commercially viable due to the added disk requirements (10–20% must be added to allow for the Hamming disks).
- RAID-3—This also distributes data in small increments but adds only one parity disk. This results in good performance for large transfers, but small transfers show poor performance.
- RAID-4—In order to overcome the small transfer performance penalties in RAID-3, RAID-4 uses large data chunks distributed over several disks and a single parity disk. This results in a

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

bottleneck at the parity disk. Due to this performance problem RAID-4 is not considered commercially viable.

- RAID-5—This solves the bottleneck by distributing the parity data across the disk array. The major problem is it requires several write operations to update parity data. The performance hit is only moderate and the other benefits outweigh this minor problem.
- RAID-6—This adds a second redundancy disk that contains error-correction codes. Read performance is good due to load balancing, but write performance suffers due to RAID-6 requiring more writes than RAID-5 for data update.

For the money, I would suggest RAID0/1, that is, striped and mirrored. It provides nearly all of the dependability of RAID5 and gives much better write performance. You will usually take at least a 20% write performance hit using RAID5. For read-only applications RAID5 is a good choice, but in high transaction/high performance environments the write penalties may be too high.

New Technologies

Oracle is a broad topic; topics related to Oracle and Oracle data storage are even broader. This section will touch on several new technologies such as Optical Disk, RAM disk, and tape systems that should be utilized with Oracle systems whenever possible. Proper use of Optical technology can result in significant savings when large volumes of static data are in use in the database (read only). RAM drives can speed access to index and small table data by several-fold. High-speed tapes can make backup and recovery go quickly and easily. Let's examine these areas in more detail.

Optical Disk Systems

WORM (write once, read many) or MWMR (multiple write, multiple read) optical disks can be used to great advantage in an Oracle system. Their main use will be in storage of export and archive log

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

files. Their relative immunity to crashes and long shelf life provide an ideal solution to the storage of the immense amount of data that proper use of archive logging and exports produce. As access speeds improve, these devices will be worth considering for these applications in respect to Oracle. Another area where they have shown great benefits is in read-only tablespaces. Now in Oracle8I with transportable tablespaces it becomes possible to create an entire catalog system on one Oracle server, place the tablespaces on CD-ROMs or PDCD-ROMs and literally ship copies to all of your sites where they will up and operating the day they get there.

Tape Systems

Nine track, 4 mm, 8 mm, and the infamous TK series from DEC can be used to provide a medium for archive logs and exports. One problem with this is the need at most installations for operator monitoring of the tape devices to switch cartridges and reels. With the event of stacker-loader drives for the cartridge tapes, this limitation has all but been eliminated in all but the smallest shops. New DAT tape technology with fast streaming tape makes for even faster backup and recovery times.

RAM Drives (Random Access Memory)

While RAM drives have been around for several years, they have not seen the popularity their speed and reliability should be able to claim. One of the problems has been their small capacity in comparison to other storage mediums. Several manufacturers offer solid state drives of steadily increasing capacities. For index storage these devices are excellent. Their major strength is their innate speed. They also have onboard battery backup sufficient to back up their contents to their built-in hard drives. This backup is an automatic procedure invisible to the user, as is the reload of data upon power restoration. The major drawback to RAM drives is their high cost. The rapid reductions in memory chip costs with the equally rapid increase in amount of storage per chip may soon render this drawback nonexistent.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

New disk arrays such as those developed by EMC Technology provide a hybrid between disk and RAM technology with their multi-gigabyte high reliability arrays and multi-gigabyte RAM caches.

Backup & recovery

As should be obvious from the previous lesson, Oracle is a complex, interrelated set of files and executables. With Oracle8 and Oracle8i it hasn't gotten any simpler. The database files include data segments, redo logs, rollback segments, control files, bfiles, libraries, and system areas. Each of these files is not a separate entity but is tightly linked to the others. For instance, the data files are repositories for all table data; the data file structure is controlled by the control file, implemented by the system areas, and maintained by a combination of the executables, redo, and rollback segments. Data files reference bfiles that are tied to external procedures stored in libraries that are referenced in procedures stored in data files. This complexity leads to the requirement of a threefold backup recovery methodology to ensure that data recovery can be made.

The threefold recovery methodology consists of:

1. Normal backups using system backups, Oracle Backup Manager, Recovery Manager or a third party, tested against Oracle tool.
2. Exports and imports.
3. Archive logging of redo logs.

Let's look at each of these and how they are used.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Backups

Normal system backups, referred to as either hot or cold backups, are used to protect the system from media failure. Each can and should be used when required.

Cold Backups

A cold backup, that is, one done with the database in a shutdown state, provides a complete copy of the database that can be restored exactly. The generalized procedure for using a cold backup is as follows:

1) Using the shutdown script(s) provided, shutdown the Oracle instance(s) to be backed up.
2) Ensure that there is enough backup media to back up the entire database.
3) Mount the first volume of the backup media (9 track, WORM, 4mm, 8mm, etc.) using the proper operating system mount command:

   For example on OpenVMS:

   ```
   $ mount/foreign dev: volume_name
   ```

   On UNIX:

   ```
   $ umount –orw /dev/rmt0 /tape1
   ```

4) Issue the proper Operating System backup command to initiate the backup.

   For example on OpenVMS:

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
$ backup/log=ora_<date>.log -
$ ora_diskx:[oracle...]*.*,-
$ ora_diskx+1:[oracle...]*.* -
$ ora_diskx+2:[oracle...]*.*
$ ora_diskx+n:oracle...] dev:ora_<date>.sav/save
```

Where:

- ora_diskx. . . . ora_diskn represents the system logicals in accordance with OFA rules, with n being the highest numbered disk and x being the lowest.
- <date> represents the date for the backup.
- dev: represents the backup media device name such as mua0:
- /log=log_<date>.log names a file to log the results from the backup.
- /save this tells BACKUP to archive the files in save set format; this requires less room than an image backup.

On UNIX:

```
$ tar –cvf /tape1 /ud*/oracle*/ortest1/*
```

(for all Oracle data, log and trace files assuming an OFA installation)

Where:

- tar – this is short for tape archiver and is the default backup command on UNIX, raw volumes may require "dd"
- -cvf – These arguments tell tar to c: create a new archive, v: tell us what it is doing, f: use the device specification which follows (we could have not specified a device and it would default to the default tape drive)

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

5) Once the backup is complete, be sure all backup volumes are properly labeled and stored, *away from the computer*. The final volume is dismounted from the tape drive using the appropriate operating system DISMOUNT command:

   For example on OpenVMS:

```
$ DISMOUNT dev:
On UNIX:
$ umount /tape1
```

6) Restart the Oracle instances using the appropriate startup script(s).

Hot Backups

A hot backup, or one taken while the database is active, can only give a read-consistent copy but doesn't handle active transactions. You must ensure that all redo logs archived during the backup process are also backed up. The hot backup differs from the cold backup in that only sections of the database are backed up at one time. This is accomplished by using the ALTER command to modify a tablespace's status to backup. Be sure that you restore the status to normal once the database is backed up or else redo log mismatch and improper archiving/rollbacks can occur.

While it is quite simple (generally speaking) to do a cold backup by hand, a hot backup can be quite complex and should be automated. The automated procedure should then be thoroughly tested on a dummy database for both proper operation and ability to restore prior to its use on the production database(s).

Limitations on hot or on-line backups:

- The database must be operating in ARCHIVELOG mode for hot backups to work.
- Hot backups should only be done during off or low-use periods.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- During the hot backups the entire block containing a changed record, not just the changed record, is written to the archive log, requiring more archive space for this period.

The hot backup consists of three processes:

1. The tablespace data files are backed up.
2. The archived redo logs are backed up.
3. The control file is backed up.

The first two parts have to be repeated for each tablespace in the database. For small databases, this is relatively easy. For large, complex databases with files spread across several drives, this can become a nightmare if not properly automated in operating system specific command scripts.

As you can see, this is a bit more complex than a full cold backup and requires more monitoring than a cold backup. Recovery from this type of backup consists of restoring all tablespaces and logs and then recovering. You only use the backup of the control file if the current control file was also lost in the disaster; otherwise, be sure to use the most current copy of the control file for recovery operations.

TIP

In a number of computer facilities backups are kept close at hand, sometimes in the same room as the computer. What would happen if a site disaster destroyed the computer room? Not only the hardware, but all of the system backups and your data could be lost. Store backups in another building or even totally off-site somewhere. This assures that come fire, flood, or typhoon, you should be able to get backup one way or another.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

UNIX of course requires a different scripting language and command set.

One problem with a canned script for hot backup is that they don't automatically reconfigure themselves to include new tablespaces, or redo logs. The script shown below is an example of how to let Oracle build its own hot backup script using dynamic SQL and the data dictionary. This script is excerpted from `Oracle Administrator` from RevealNet, Inc.Version 99.2 (an online reference product).

SOURCE 1 Example of script to generate a hot backup script on UNIX.

```
REM Script to create a hot backup script on UNIX
REM Created 6/23/98 MRA
REM
create table bu_temp (line_no number,line_txt
varchar2(2000))
storage (initial 1m next 1m pctincrease 0);
truncate table bu_temp;
set verify off embedded off
define dest_dir=&1;
declare
--
-- Declare cursors
--
-- Cursor to get all tablespace names
--
cursor get_tbsp is
select tablespace_name from dba_tablespaces;
--
-- cursor to create BEGIN BACKUP command
--
cursor bbu_com (tbsp varchar2) is
select
'alter tablespace '||tablespace_name||' begin backup;'
from dba_tablespaces where tablespace_name=tbsp;
--
-- Cursor to create HOST backup commands
--
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
cursor tar1_com (tbsp varchar2) is
select '/bin/tar cvf - '||file_name||' \'
from dba_data_files where tablespace_name=tbsp
and file_id=(select min(file_id)from dba_data_files
where tablespace_name=tbsp);
--
cursor tar2_com (tbsp varchar2) is
select
file_name||' \'
from dba_data_files where tablespace_name=tbsp
and file_id>(select min(file_id) from dba_data_files
where tablespace_name=tbsp);
--
cursor tar3_com (tbsp varchar2) is
select '/bin/tar cvrf - '||file_name||' \'
from dba_data_files where tablespace_name=tbsp
and file_id=(select min(file_id)from dba_data_files
where tablespace_name=tbsp);
--
cursor comp_com (tbsp varchar2) is
select
'|compress>&&dest_dir/'||tablespace_name||'_'||to_char(sysd
ate,'dd_mon_yy')||'.Z'||chr(10)||'exit'
from dba_tablespaces where tablespace_name=tbsp;

--
-- Cursor to create END BACKUP command
--
cursor ebu_com (tbsp varchar2) is
select
'alter tablespace '||tablespace_name||' end backup;' from
dba_tablespaces
where tablespace_name=tbsp;
--
-- Cursor to create redo log HOST backup commands
--
cursor tar1_rdo is
select '/bin/tar cvrf -  \'
from dual;
--
cursor tar2_rdo is
select
member||' \'
from v$logfile;
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
--
cursor comp_rdo is
select
'|compress>&&dest_dir/redo_logs_'||to_char(sysdate,'dd_mon_
yy')||'.Z'||chr(10)||'exit'
from dual;
--
-- Temporary variable declarations
--
tbsp_name varchar2(64);
line_num number:=0;
line_text varchar2(2000);
min_value number;
first_tbsp boolean;
--
-- Begin build of commands into temporary table
--
begin
--
-- first, create script header
--
line_num := line_num+1;
select 'REM Online Backup Script for '||name||' instance'
into line_text from v$database;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'REM Script uses UNIX tar format backup commands'
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'REM created on '||to_char(sysdate, 'dd-mon-yyyy
hh24:mi')||' by user '||user
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'REM developed for RevealNet by Mike Ault - DMR
Consulting Group 7-Oct-1998'
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'REM Script expects to be fed backup directory
location on execution.'
into line_text from dual;
insert into bu_temp values (line_num,line_text);
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
line_num := line_num+1;
select 'REM Script should be re-run anytime physical
structure of database altered.'
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'REM '
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
--
-- Now get tablespace names and loop through until all are
handled
--
open get_tbsp;
first_tbsp:=TRUE;
loop
--
-- Get name
--
      fetch get_tbsp into tbsp_name;
      exit when get_tbsp%NOTFOUND;
--
-- Add comments to script showing which tablespace
--
      select 'REM' into line_text from dual;
      insert into bu_temp values (line_num,line_text);
      line_num:=line_num+1;
      select 'REM Backup for tablespace '||tbsp_name into
line_text from dual;
      insert into bu_temp values (line_num,line_text);
      line_num:=line_num+1;
      select 'REM' into line_text from dual;
      insert into bu_temp values (line_num,line_text);
      line_num:=line_num+1;
--
-- Get begin backup command built for this tablespace
--
      open bbu_com (tbsp_name);
      fetch bbu_com into line_text;
      insert into bu_temp values (line_num,line_text);
      line_num:=line_num+1;
      close bbu_com;
--
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
-- The actual backup commands are per datafile, open cursor
and loop
--
      open tar1_com (tbsp_name);
      open tar2_com (tbsp_name);
open tar3_com (tbsp_name);
      open comp_com (tbsp_name);
      min_value:=line_num;
      loop
            if line_num=min_value
            then
                        select 'host' into line_text from
dual;
                        insert into bu_temp values
(line_num,line_text);
                        line_num:=line_num+1;
                  if first_tbsp THEN
                  fetch tar1_com into line_text;
                else
                  fetch tar3_com into line_text;
                end if;
            else
                  fetch tar2_com into line_text;
                        exit when tar2_com%NOTFOUND;
            end if;
            insert into bu_temp values
(line_num,line_text);
            line_num:=line_num+1;
           first_tbsp:=FALSE;
      end loop;
      fetch comp_com into line_text;
      insert into bu_temp values (line_num,line_text);
    line_num:=line_num+1;
      close tar1_com;
      close tar2_com;
close tar3_com;
      close comp_com;
--
-- Build end backup command for this tablespace
--
  open ebu_com(tbsp_name);
  fetch ebu_com into line_text;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
  close ebu_com;
end loop;
  close get_tbsp;
--
-- Backup redo logs, normally you won't recover redo logs
you
-- will use your current redo logs so current SCN
information not lost
-- commands just here for completeness
--
  select 'REM' into line_text from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'REM Backup for redo logs' into line_text from
dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'REM Normally you will not recover redo logs' into
line_text from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'REM' into line_text from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
--
-- Create host backup commands for all redo logs
--
  open tar1_rdo;
  open tar2_rdo;
  open comp_rdo;
  min_value:=line_num;
  loop
      if line_num=min_value
      then
       select 'host' into line_text from dual;
       insert into bu_temp values (line_num,line_text);
       line_num:=line_num+1;
       fetch tar1_rdo into line_text;
      else
       fetch tar2_rdo into line_text;
       exit when tar2_rdo%NOTFOUND;
      end if;
      insert into bu_temp values (line_num,line_text);
      line_num:=line_num+1;
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
  end loop;
  fetch comp_rdo into line_text;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  close tar1_rdo;
  close tar2_rdo;
  close comp_rdo;
--
-- Now get all archive logs, performing a switch to be sure
all
-- required archives are written out
--
  select 'REM' into line_text from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'REM Backup for archive logs' into line_text from
dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'REM' into line_text from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'alter system switch logfile;' into line_text from
dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'archive log all;' into line_text from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
--
-- The next command builds the actual backup command based
on the
-- value of the log_archive_dest initialization parameter,
it looks for the
-- last right square bracket in the name and just uses that
section with
-- a wildcard
--
  select 'host compress '||substr
(value,1,instr(value,'/',-1,1))||'*'
  into line_text from v$parameter where
name='log_archive_dest';
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
  select 'host tar cvrf - '||substr
(value,1,instr(value,'/',-1,1))||'*.Z'||
  '|compress>&&dest_dir/'||
  substr (value,instr(value,'/',-
1,1)+1,length(value))||'_'||to_char(sysdate,'dd_mon_yy')||'
.Z'
  into line_text from v$parameter where
name='log_archive_dest';
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
--
-- Next, backup a control file just to be sure
-- we have a good one available that is current with this
backup
--
   select 'alter database backup control file to
'||'&&dest_dir'||'/ora_conbackup_'||to_char(sysdate,'dd_mon
_yy')||'.bac;'
   into line_text from dual;
   insert into bu_temp values (line_num,line_text);
   line_num:=line_num+1;
end;
/
rem
rem Now generate output based on bu_temp table contents
rem
set verify off feedback off heading off termout off pages 0
set embedded on lines 132
column line_no noprint
column dbname new_value db noprint
select name dbname from v$database;
spool rep_out\&db\thot_bu.sql
select * from bu_temp order by line_no;
spool off
rem directory syntax for UNIX
rem
rem host  sed '1,$ s/ *$//g' thot_bu.sql>hot_bu.sql
rem
drop table bu_temp;
set verify on feedback on heading on termout on pages 22
set embedded off lines 80
clear columns
undef dest_dir
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The output from the script for my test database is shown in Source 2.

```
REM Online Backup Script for ORTEST1 instance
REM Script uses UNIX tar format backup commands
REM created on 25-may-1999 20:09 by user SYSTEM
REM developed for RevealNet by Mike Ault-DBMentors
International 7-Oct-1998
REM Script expects to be fed backup directory location on
execution.
REM Script should be re-run when physical database
structure altered.
REM
REM
REM Backup for tablespace SYSTEM
REM
alter tablespace SYSTEM begin backup;
host
/bin/tar cvf - /ud2/ORACLE2/ORTEST1/DATA/SYS1ORTEST1.DBF \
|compress>/tape1/SYSTEM_25_may_99.Z
exit

alter tablespace SYSTEM end backup;
REM
REM Backup for tablespace USER_DATA
REM
alter tablespace USER_DATA begin backup;
host
/bin/tar cvrf - /ud2/ORACLE2/ORTEST1/DATA/USR1ORTEST1.DBF \
|compress>/tape1/USER_DATA_25_may_99.Z
exit

alter tablespace USER_DATA end backup;
REM
REM Backup for tablespace TOOLS
REM
alter tablespace TOOLS begin backup;
host
/bin/tar cvrf - /ud2/ORACLE2/ORTEST1/DATA/TOOLS1ORTEST1.DBF
\
|compress>/tape1/TOOLS_25_may_99.Z
exit

alter tablespace TOOLS end backup;
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
REM
REM Backup for tablespace RBS
REM
alter tablespace RBS begin backup;
host
/bin/tar cvrf - /ud3/ORACLE3/ORTEST1/DATA/RBS1ORTEST1.DBF \
|compress>/tape1/RBS_25_may_99.Z
exit

alter tablespace RBS end backup;
REM
REM Backup for tablespace TEMP
REM
alter tablespace TEMP begin backup;
host
/bin/tar cvrf - /ud4/ORACLE4/ORTEST1/DATA/TMP1ORTEST1.DBF \
/ud4/ORACLE4/ORTEST1/DATA/TMP2ORTEST1.DBF \
|compress>/tape1/TEMP_25_may_99.Z
exit

alter tablespace TEMP end backup;
REM
REM Backup for tablespace TELE_DATA
REM
alter tablespace TELE_DATA begin backup;
host
/bin/tar cvrf -
/ud2/ORACLE2/ORTEST1/DATA/TELEDAT1ORTEST1.DBF \
|compress>/tape1/TELE_DATA_25_may_99.Z
exit

alter tablespace TELE_DATA end backup;
REM
REM Backup for tablespace TELE_INDEX
REM
alter tablespace TELE_INDEX begin backup;
host
/bin/tar cvrf -
/ud3/ORACLE3/ORTEST1/DATA/TELEIDX1ORTEST1.ORA \
|compress>/tape1/TELE_INDEX_25_may_99.Z
exit

alter tablespace TELE_INDEX end backup;
REM
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
REM Backup for tablespace GRAPHICS_DATA
REM
alter tablespace GRAPHICS_DATA begin backup;
host
/bin/tar cvrf -
/ud2/ORACLE2/ORTEST1/DATA/GRPHDAT1ORTEST1.DBF \
|compress>/tape1/GRAPHICS_DATA_25_may_99.Z
exit

alter tablespace GRAPHICS_DATA end backup;
REM
REM Backup for tablespace GRAPHICS_INDEX
REM
alter tablespace GRAPHICS_INDEX begin backup;
host
/bin/tar cvf -
/ud3/ORACLE3/ORTEST1/DATA/GRPHIDX1ORTEST1.ORA \
|compress>/tape1/GRAPHICS_INDEX_25_may_99.Z
exit

alter tablespace GRAPHICS_INDEX end backup;
REM
REM Backup for redo logs
REM Normally you will not recover redo logs
REM
host
/bin/tar cvrf -  \
/ud3/ORACLE3/ORTEST1/REDO/LOG11ORTEST1.DBF \
/ud4/ORACLE4/ORTEST1/REDO/LOG12ORTEST1.DBF \
/ud4/ORACLE4/ORTEST1/REDO/LOG21ORTEST1.DBF \
/ud1/ORACLE1/ORTEST1/REDO/LOG22ORTEST1.DBF \
/ud1/ORACLE1/ORTEST1/REDO/LOG31ORTEST1.DBF \
/ud3/ORACLE3/ORTEST1/REDO/LOG32ORTEST1.DBF \
|compress>/tape1/redo_logs_25_may_99.Z
exit

REM
REM Backup for archive logs
REM
alter system switch logfile;
archive log all;
host compress /ud4/ORACLE4/ORTEST1/ARCHIVE/*
host tar cvrf - *.Z|compress>/tape1/_25_may_99.Z
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
alter database backup control file to
/tape1/ora_conbackup_25_may_99.bac;
```

SOURCE 2 Example of output from the hot backup script generator.

A similar script for and NT is shown in Source 3. You will need to verify that the target directories exist or modify the scripts before running them. The NT script assumes a backup staging area is being used that is then backed up to tape.

```
REM Script to create a hot backup script on NT using ocopy
REM Created 6/23/98 MRA
REM
REM Get oracle home path and backup file destination
REM &&ora_home
REM &&dest_dir
REM create holding table for commands
REM
create table bu_temp (line_no number,line_txt
varchar2(2000));
REM
REM Empty it if it already exists
REM
truncate table bu_temp;
REM
REM
set verify off embedded off
set esc ^
REM
REM Set backup command
REM
column dup new_value dup_it noprint
select ''||chr(39)||'host start/wait
'||'&&ora_home'||'\bin\ocopy '||chr(39)||'' dup
from dual;
REM
REM Begin script creation
REM
declare
--
-- Declare cursors
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
--
-- Cursor to get all tablespace names
--
cursor get_tbsp is
select tablespace_name from dba_tablespaces;
--
-- cursor to create BEGIN BACKUP command
--
cursor bbu_com (tbsp varchar2) is
select
'alter tablespace '||tablespace_name||' begin backup;'
from dba_tablespaces where tablespace_name=tbsp;
--
-- Cursor to create HOST backup commands
--
cursor bu_com (tbsp varchar2) is
select
&&dup_it||file_name||'
'||'&&dest_dir'||'\datafiles\'||tbsp||file_id||'.bck'
from dba_data_files where tablespace_name=tbsp;
--
-- Cursor to create END BACKUP command
--
cursor ebu_com (tbsp varchar2) is
select
'alter tablespace '||tablespace_name||' end
backup;'||chr(10) from
dba_tablespaces
where tablespace_name=tbsp;
--
-- Cursor to create redo log HOST backup commands
--
cursor bu_rdo is
select
&&dup_it||member||' '||
'
'||'&&dest_dir'||'\logs'||substr(member,instr(member,'\LOG'
,2,1),instr(member,'.',1,1))
from v$logfile order by group#;
--
-- Temporary variable declarations
--
tbsp_name varchar2(64);
line_num number:=0;
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
line_text varchar2(2000);
--
-- Begin build of commands into temporary table
--
begin
--
-- first, create script header
--
line_num := line_num+1;
select 'set echo on'
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'spool
'||'&&dest_dir'||'\bu'||to_char(sysdate,'ddmonyy')||'.log'
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'REM Online Backup Script for '||name||' instance'
into line_text from v$database;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'REM Script uses ocopy - NT format backup commands'
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'REM created on '||to_char(sysdate, 'dd-mon-yyyy
hh24:mi')||' by user '||user
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'REM developed for RevealNet by Mike Ault - DMR
Consulting Group 15-Dec-1998'
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'REM '
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'REM Script should be re-run anytime physical
structure of database altered.'
into line_text from dual;
insert into bu_temp values (line_num,line_text);
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
line_num := line_num+1;
select 'REM '
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
--
-- Now get tablespace names and loop through until all are
handled
--
open get_tbsp;
loop
--
-- Get name
--
      fetch get_tbsp into tbsp_name;
      exit when get_tbsp%NOTFOUND;
--
-- Add comments to script showing which tablespace
--
      select 'REM' into line_text from dual;
      insert into bu_temp values (line_num,line_text);
      line_num:=line_num+1;
      select 'prompt Backup for tablespace '||tbsp_name
into line_text from dual;
      insert into bu_temp values (line_num,line_text);
      line_num:=line_num+1;
      select 'REM' into line_text from dual;
      insert into bu_temp values (line_num,line_text);
      line_num:=line_num+1;
--
-- Get begin backup command built for this tablespace
--
      open bbu_com (tbsp_name);
      fetch bbu_com into line_text;
      insert into bu_temp values (line_num,line_text);
      close bbu_com;
--
-- The actual backup commands are per datafile, open cursor
and loop
--
      open bu_com (tbsp_name);
      loop
                fetch bu_com into line_text;
            exit when bu_com%NOTFOUND;
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
            line_num:=line_num+1;
            insert into bu_temp values
(line_num,line_text);
      end loop;
      close bu_com;
--
-- Build end backup command for this tablespace
--
  open ebu_com(tbsp_name);
  fetch ebu_com into line_text;
  line_num := line_num+1;
  insert into bu_temp values (line_num,line_text);
  close ebu_com;
  line_num := line_num+1;
  select ' ' into line_text from dual;
  insert into bu_temp values (line_num,line_text);
end loop;
  close get_tbsp;
--
-- Backup redo logs, normally you won't recover redo logs
you
-- will use your current redo logs so current SCN
information not lost
-- commands just here for completeness
--
  select 'REM' into line_text from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'prompt Backup for redo logs' into line_text from
dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'REM Normally you will not recover redo logs' into
line_text from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'REM' into line_text from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
--
-- Create host backup commands for all redo logs
--
  open bu_rdo;
  loop
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
      fetch bu_rdo into line_text;
      exit when bu_rdo%NOTFOUND;
      line_num:=line_num+1;
      insert into bu_temp values (line_num,line_text);
  end loop;
  close bu_rdo;
--
-- Now get all archive logs, performing a switch to be sure
all
-- required archives are written out
--
  line_num:=line_num+1;
  select 'REM' into line_text from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'prompt Backup for archive logs' into line_text
from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'REM' into line_text from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'alter system switch logfile;' into line_text from
dual;
  line_num:=line_num+1;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'alter system archive log all;' into line_text
from dual;
  line_num:=line_num+1;
  insert into bu_temp values (line_num,line_text);
--
-- The next command builds the actual backup command based
on the
-- value of the log_archive_dest initialization parameter,
it looks for the
-- last right square bracket in the name and just uses that
section with
-- a wildcard.
--
  select &&dup_it||value||'\*.*
'||'&&dest_dir'||'\archives\*.*'
  into line_text from v$parameter where
name='log_archive_dest';
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
  line_num:=line_num+1;
  insert into bu_temp values (line_num,line_text);
--
-- Next, backup a control file just to be sure
-- we have a good one available that is current with this
backup
--
   select 'alter database backup controlfile to
'||chr(39)||'&&dest_dir'||'\ora_conbackup.bac'||chr(39)||';
'
   into line_text from dual;
   line_num:=line_num+1;
   insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'spool off'
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'exit'||CHR(10)
into line_text from dual;
insert into bu_temp values (line_num,line_text);
end;
/
rem
rem Now generate output based on bu_temp table contents
rem
set verify off feedback off heading off termout off pages 0
set embedded on lines 132
column db_name new_value db noprint
column line_no noprint
select name db_name from v$database;
spool rep_out/&&db/thot_bu.sql
select * from bu_temp order by line_no;
spool off
rem
rem get rid of bu_temp table
rem
drop table bu_temp;
set verify on feedback on heading on termout on pages 22
set embedded off lines 80
clear columns
undef dest_dir
undef ora_home
exit
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Source 3 Example NT Oracle Hot Backup Script Generator

I suggest that at the same time the backup script is generated, the recovery script also be generated. Source 4 shows an example recovery script generator for Nt.

```
REM Script to create a hot backup recovery script on NT
using ocopy
REM Created 6/23/98 MRA
REM
create table bu_temp (line_no number,line_txt
varchar2(2000));
truncate table bu_temp;
set verify off embedded off esc ^
REM &&ora_home &&dest_dir
column dup new_value dup_it noprint
select ''||chr(39)||'&&ora_home'||'\ocopy '||chr(39)||''
dup
from dual;
--
declare
--
-- Declare cursors
--
-- Cursor to get all tablespace names
--
cursor get_tbsp is
select tablespace_name from dba_tablespaces;
--
-- Cursor to create recovery commands
--
cursor rec_com (tbsp varchar2) is
select
&&dup_it||'
'||'&&dest_dir'||'\datafiles\'||tbsp||file_id||'.bck
'||file_name
from dba_data_files where tablespace_name=tbsp;
--
-- Cursor to create redo log recovery commands
--
cursor rec_rdo (num number) is
select
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
&&dup_it||
'
'||'&&dest_dir'||'\logs'||substr(member,instr(member,'\LOG'
,2,1),instr(member,'.',1,1))||' '||
member
from v$logfile order by group#;
--
-- Temporary variable declarations
--
tbsp_name varchar2(64);
line_num number:=0;
line_text varchar2(2000);
num number:=0;
--
-- Begin build of commands into temporary table
--
begin
--
-- first, create script header
--
line_num := line_num+1;
select 'REM Recovery Script for '||name||' instance'
into line_text from v$database;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'REM Script uses ocopy - NT format backup commands'
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'REM created on '||to_char(sysdate, 'dd-mon-yyyy
hh24:mi')||' by user '||user
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'REM developed for RevealNet by Mike Ault - DMR
Consulting 15-Dec-1998'
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'REM '
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
select 'REM Script should be re-run anytime physical
structure of database altered.'
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
select 'REM '
into line_text from dual;
insert into bu_temp values (line_num,line_text);
line_num := line_num+1;
--
-- Now get tablespace names and loop through until all are
handled
--
open get_tbsp;
loop
--
-- Get name
--
      fetch get_tbsp into tbsp_name;
      exit when get_tbsp%NOTFOUND;
--
-- Add comments to script showing which tablespace
--
      select 'REM' into line_text from dual;
      insert into bu_temp values (line_num,line_text);
      line_num:=line_num+1;
      select 'REM Recovery for tablespace '||tbsp_name into
line_text from dual;
      insert into bu_temp values (line_num,line_text);
      line_num:=line_num+1;
      select 'REM' into line_text from dual;
      insert into bu_temp values (line_num,line_text);
      line_num:=line_num+1;
--
-- The actual recovery commands are per datafile, open
cursor and loop
--
      open rec_com (tbsp_name);
      loop
                fetch rec_com into line_text;
            exit when rec_com%NOTFOUND;
            line_num:=line_num+1;
            insert into bu_temp values
(line_num,line_text);
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
      end loop;
      close rec_com;
end loop;
  close get_tbsp;
--
-- Recover redo logs, normally you won't recover redo logs
you
-- will use your current redo logs so current SCN
information not lost
-- commands just here for completeness uncomment commands
below to
-- enable redo log recovery (not advised)
--
  select 'REM' into line_text from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'REM Recovery for redo logs' into line_text from
dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'REM Normally you will not recover redo logs' into
line_text from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'REM' into line_text from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
--
-- Create host backup commands for all redo logs
--
  /*open rec_rdo(num);
  loop
      fetch rec_rdo into line_text;
      exit when rec_rdo%NOTFOUND;
      num:=num+1;
      line_num:=line_num+1;
      insert into bu_temp values (line_num,line_text);
  end loop;
  close rec_rdo;*/
--
-- Now recover all archive logs
--
  line_num:=line_num+1;
  select 'REM' into line_text from dual;
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'REM Recovery for archive logs' into line_text
from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
  select 'REM' into line_text from dual;
  insert into bu_temp values (line_num,line_text);
  line_num:=line_num+1;
--
-- The next command builds the actual recovery command
based on the
-- value of the log_archive_dest initialization parameter,
it looks for the
-- last right square bracket in the name and just uses that
section with
-- a wildcard
--
  select &&dup_it||' '||'&&dest_dir'||'\archives\*.*
'||value||'\*.*'
  into line_text from v$parameter where
name='log_archive_dest';
  line_num:=line_num+1;
  insert into bu_temp values (line_num,line_text);
end;
/
rem
rem Now generate output based on bu_temp table contents
rem
set verify off feedback off heading off termout off pages 0
set embedded on lines 132
column db_name new_value db noprint
column line_no noprint
select name db_name from v$database;
spool rep_out\&&db\rec_db.bat
select * from bu_temp order by line_no;
spool off
rem
rem get rid of bu_temp table
rem
drop table bu_temp;
set verify on feedback on heading on termout on pages 22
set embedded off lines 80 esc \
clear columns
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
undef ora_home
undef dest_dir
exit
```

Source 4 Example Recovery Script Generator for NT

Once you have generated the scripts to generate the online backup and recovery files, document them. The next section is an example documentation procedure for the NT online backup and recovery scripts.

Example Documentation Procedure for NT Online Backup and Recovery Scripts

This section shows an example set of procedures for use of the NT Oracle hot backup and recovery
Scripts:

Backup:

1. Run nt_oline_bu.sql from SQLPLUS DBA account.
2. Run nt_rec_db.sql script from SQLPLUS DBA account.
3. Move a copy of the rec_db.bat script generated in step 2 to the backup directory.
4. From a SQLPLUS command line session (using the e:\orant81\bin\sqlplus executable) run the thot_bu.sql script generated in step 1.
5. Once step 4 completes (should be less than x hours) copy the backup directory (I:\backup) using the system backup tool, to tape.
6. Remove the archive logs that were copied from the database archive log destination to tape from the archive log destination.

Recovery:

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

1. Using the system backup tools, restore the Oracle backup files to the backup location on the database server (Example: I:\backup).
2. Run the recovered copy of the rec_db.bat script to restore the backup files to their proper locations.
3. Manually start the oracle services and the tns listener process using the control panel services icon.
4. From the command line, use the svrmgrl executable to startup and mount (but not open) the database:

   >svrmgrl

   svrmgrl>connect internal@<alias>.world

   password: xxxxxxxxx

   connected to an idle instance

   svrmgrl>startup mount pfile=e:\orant\database\init<SID>.ora
   (Be sure to use the location of your initialization file)

   <will see normal startup messages>

   svrmgrl> recover

   <server will prompt for needed files, they should be already copied to machine so just press return at each prompting>

   media recovery complete

   svrmgrl> alter database open

   database altered
5. Shutdown and perform a cold backup of all database files (essentially take the ocopy commands from inside the thot_bu.sql script and run them as a .bat file. Do not backup the archive logs, after a cold backup they are not needed anymore.
6. Remove all archive logs from system
7. Database is recovered, resume normal operations.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The actual backup process can be automated on NT using the WINAT scheduler available from the Microsoft website or the Microsoft support or toolkit CD-ROM. A script similar to the one shown in Source 5 should be used to start the backup.

```
REM do_hot_bu.bat
REM File to generate and execute hot backup script for
Oracle
REM Used for ORTEST1 database only
REM Mike Ault DBMentors International
REM
REM First, generate the thot_bu.sql script
REM
cd c:\sql_scripts
REM
e:\orant81\bin\sqlplus -s system/manager@ortest1.world
@c:\sql_scripts\nt_oline_bu.sql
REM
REM Now generate the recovery script so they are in-sync
REM
e:\orant81\bin\sqlplus -s system/manager@ortest1.world
@c:\dmr_temp\nt_rec_db.sql
REM
REM Copy the recovery script to the backup destination
REM
copy c:\sql_scripts\rep_out\ortest1\rec_db.bat
i:\backup\rec_db.bat
REM
REM Run the backup script
REM
e:\orant81\bin\sqlplus -s system/manager@ortest1.world
@c:\sql_scripts\rep_out\ortest1\thot_bu.sql
REM
REM End of script
REM
Exit
```

Source 5 Example NT .bat Script To Start Backup

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Imports/Exports

Imports and exports extract or insert an Oracle-readable copy of the actual data and structures in the database. The exports can be used to recover single data structures to the date and time the export was taken. Exports come in three types: full, cumulative, and incremental. Full, as its name implies, provides a full logical copy of the database and its structures. A cumulative provides a complete copy of altered structures since the last full or the last cumulative export. Incremental exports provide a complete copy of altered structures since the last incremental, cumulative, or full export.

Limitations on export/import:

- A database must be running to perform either an export or import.
- Export files shouldn't be edited and can only be used by import.
- (Import only) imports full tables; it can't be used to do a conditional load.
- Exported data is only a logical copy of the data. An export can only allow recovery to the date and time the export was taken.
- Imports and exports are accomplished using the Oracle IMPORT and EXPORT utilities.

Exports

For exports, the EXPORT utility is used. The format for using this command follows:

Format: EXP KEYWORD=value —or— KEYWORD=(list of values)

Example: EXP AULT/AUTHOR GRANTS=N TABLES=(CHAPTERS, EDITORS,ADVANCES)

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Keyword	Description (Default)
USERID	username/password
BUFFER	size of data buffer
FILE	output file (EXPDAT.DMP)
COMPRESS	import into one extent (Y)
GRANTS	export grants (Y)
INDEXES	export indexes(Y)
ROWS	export data rows (Y)
CONSTRAINTS	export table constraints (Y)
CONSISTENT	cross-table consistency (N)
LOG	log file of screen output (None)
STATISTICS	analyze objects (ESTIMATE)
DIRECT	Bypass the SQLcommand processing layer (N) (new in Oracle8)
FEEDBACK	Show a process meter (a dot) every X rows exported (0 – Xvalue)
HELP	Shows help listing
MLS	MLS_LABEL_FORMAT Used with secure Oracle; we won't cover these.
FULL	export entire file (N)
OWNER	list of owner usernames
TABLES	list of table names
RECORDLENGTH	length of IO record
INCTYPE	incremental export type
RECORD	track incr. export (Y)
PARFILE	parameter file name

Exports should be automated and scheduled to run automatically. An export methodology should be worked out such that the DBA is reasonably certain a deleted file can be recovered. The parameters for export can either be placed on the command line, or, in a parameter file which can then be accessed using the PARFILE command line option.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

IMPORT

The format of the IMPORT command follows.

Format: IMP KEYWORD=value —or— KEYWORD=(list of values)

Example: IMP AULT/AUTHOR IGNORE=Y TABLES=(EXPENSES, ADVANCES) FULL=N

Keyword	Description (Default)
USERID	username/password
BUFFER	size of data buffer
FILE	Output file (EXPDAT.DMP)
SHOW	just list file contents (N)
IGNORE	Ignore create errors (N)
RECORDLENGTH	length of IOrecord
GRANTS	Import grants (Y)
INDEXES	Import indexes (Y)
ROWS	Import data rows (Y)
LOG	log file of screen output
INDEXFILE	write table/index info to specified file
FULL	Import entire file (N)
FROMUSER	list of owner usernames
TOUSER	list of usernames
TABLES	list of table names
FEEDBACK	Provide dot status graph (0)
INCTYPE	incremental import type
COMMIT	commit array insert (N)
PARFILE	parameter file name
DESTROY	overwrite tablespace data (N)
CHARSET	char. set of export file (NLS_LANG)

Under Oracle7, the user must be granted the EXP_FULL_DATABASE role in order to do full exports. In order to perform a full import, the user must have the

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

IMP_FULL_DATABASE role. The users with the DBA role are granted these implicitly.

An example of when the DBA would want to grant these roles to a user would be a user whose password is specified in the command script used for doing the automatic exports. If the only role granted to the user is CREATE_SESSION and EXP_FULL_DATABASE even if the user's password is compromised, they won't be able to do much damage.

Archive Logs

The redo logs store all transactions that alter the database, all committed updates, adds, or deletes of tables, structures, or data. If archiving is disabled, only data in the current off-line and on-line redo logs can be recovered. If the system recycles through all redo logs, the old ones are reused, destroying their contents. If archive logging is enabled, the redo logs are written out to storage before reuse. Archive logging allows recovery to a specific point in time since the last full cold backup or complete off-line backup. Under Oracle8i archive logs can be duplexed. The initialization parameters that control archive logging are:

Parameter	Meaning
LOG_ARCHIVE_START	If set to TRUE start archive process
LOG_ARCHIVE_BUFFERS	Number of log archive buffers
LOG_ARCHIVE_BUFFER_SIZE	Size of the log archive buffers
LOG_ARCHIVE_MIN_SUCCEED_DEST	Percentage of archive logs which must reach destinations
LOG_ARCHIVE_DEST	Primary Archive Log Location

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Meaning
LOG_ARCHIVE_DUPLEX_DEST	Secondary Archive Log Location
LOG_ARCHIVE_DEST_1	Archive tertiary location 1
LOG_ARCHIVE_DEST_2	Archive tertiary location 2
LOG_ARCHIVE_DEST_3	Archive tertiary location 3
LOG_ARCHIVE_DEST_4	Archive tertiary location 4
LOG_ARCHIVE_DEST_5	Archive tertiary location 5
LOG_ARCHIVE_FORMAT	Specifies the format for archive log names, %s and %t add the sequence and redo thread numbers to the format.

Under Oracle7 and Oracle8 redo logs are specified in groups; each group forms a shadow set and is archived together. Under Oracle7 archive logs can also be assigned to threads for use in parallel or shared instances. Individual logs are called members. Threads hold groups that hold members. Each member of a redo log group is the same size and should be on separate physical platters. Oracle automatically synchronizes members of a group into a shadow set.

Redo logs cannot be used to recover a database brought back from a full export.

To switch a database that is not currently using archive logging to use archive logging the steps are:

1. Shutdown database using immediate or normal options
2. Edit the initialization parameter file to include appropriate archive log parameters, at a minimum:

   ARCHIVE_LOG_START = TRUE

   ARCHIVE_LOG_DEST = destination (operating system specific path to archive log destination)

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

ARCHIVE_LOG_FORMAT = arch_%t_%s.arc

Usually the defaults for LOG_ARCHIVE_BUFFERS and LOG_ARCHIVE_BUFFER_SIZE are sufficient.

3. Using the appropriate interface (svrmgrl, svrmg23, sqlplus) startup the database in mounted mode:

   Svrmgrl> connect internal/password as sysdba

   Svrmgrl> startup mount pfile=<initialization file location>

4. Use the ALTER DATABASE command to reset the ARCHIVELOG mode:

   Svrmgrl> ALTER DATABASE ARCHIVELOG;

5. Use the ALTER DATABASE command to open the database.

   Svrmgrl> ALTER DATABASE OPEN;

6. Either shutdown and perform a cold backup or perform a hot backup. Since this is the first backup, I would suggest a cold backup be used. This is the baseline backup of your database.
7. Restart the database as you would normally.

Proper use of these backup/recovery tools allows the DBA to recover from any possible failure.

Backup Methodologies

NT or UNIX System Backup

NT or UNIX system backups (such as TAR or DUMP) should be taken on at least a weekly basis. The Oracle database(s) can either be shut down or a hot backup may be used and a full backup taken.

It should be remembered that recovery is only to the time the backup was taken. All control, data, and archive logs must be backed up at the same time. The data from the backup forward is

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

recovered from the archive and on-line redo logs. Therefore, even though redo logs are backed up, they should not be restored unless the on-line redo logs have been lost. Always recover using an on-line redo log and the most current control file if possible. If you must use a control file restored from the backup, use the USING BACKUP CONTROLFILE option of the recover command.

Import/Export

As a minimum, export after large table loads for major tables.

All export procedures should be logged so that they can be reviewed. Some data corruption problems, such as unreadable characters in a record, will only be caught by the export process.

Archive Logging

Archive logging is strongly suggested for all Oracle databases.

Archive logging is automatic; it may consume disk resources in a highly active environment. If recovery is required, the system will ask for the archive logs it needs and perform recovery from them. Archive logs allow point-in-time recovery. To use hot backup, archive logging is required. Some sites switch on archive logging during a hot backup and then switch it off; however, this means they can only recover to the time of the last archive log.

Using Oracle INIT.ORA parameters, the destination and frequency of the archive logging can be controlled, although the size of the redo log plays more of a role in the frequency of archive logging than any initialization parameter.

Recovery Types

Recovery depends entirely upon how the backup methodology was employed. It also depends upon what the DBA has to recover.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Recoveries can consist of:

- REDO Log Recovery—Off-line or On-line log
- ROLLBACK Segment Recovery
- Tablespace Recovery
- Table Recovery
- Recovery of the entire database
- Recovery of executables
- Recovery of control files

Each recovery may consist of one or a combination of these, depending on file placement.

Recovery generally is required only after a physical insult to the database file system. Most internal errors are corrected automatically by Oracle using the REDO and rollback logs and data buffers.

Since each site will have differing file placement and each recovery will most likely be unique, it is suggested that the *Oracle7, Oracle8 or Oracle8i Server Recovery or Backup and Recovery Manual* (Oracle, Corporation, 1999) be consulted for the appropriate recovery methodology that applies to the situation(s) in which the DBA may find him- or herself. Another good resource is Oracle Backup and Recovery by Rama Velpuri (Oracle Press, 1996).

It can't be stressed enough that backup and recovery operations should be fully proceduralized and these procedures tested frequently, especially after major database or equipment changes. An untested backup and recovery plan is no plan at all.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Oracle7 Enterprise Backup Utility

Exactly what is the Oracle/Enterprise Backup Utility (EBU) and how can it be used? Let's take a quick look at the facility and what it does for you.

The EBU facility provides an interface between the Oracle database and your backup software. It is not an interface to VMS-Backup, UNIX tar, dump, cpio, or dump. The EBU facility automatically recognizes third-party tools it is compatible with and, based on command line or script input commands, processes Oracle backups against multiple databases. EBU can make both hot (on-line) and cold (off-line) backups.

The EBU facility includes the following background processes:

- obackup—Monitors all other processes and communicates with databases, spawns
- instance manager if it isn't already operating.
- brio—Coordinates between obackup and the brdk and brtp processes. For parallel
- operations, there is one brio per parallel IO stream.
- brdk—Handles IO to disk files. There is one or more brdk processes per concurrent IO stream as controlled by the mux process.
- brtp—Handles IO to tapes, only one brtp per tape parallel IO stream (i.e., multiple tape drives used concurrently for the same backup).
- brd—(Instance Manager) This is a daemon process that monitors the backup catalog and obackup. This process also cleans up for any abnormally terminated backup operations.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The EBU facility uses a central backup repository to track backups against multiple databases. The EBU facility is invoked via the obackup command

```
obackup [[{-ch{ecksum] -t[est] }]|
-noa[utoregister] | -s[ilent] |
-ob[k_home]=path ] command_filename
```

Where

-ch Cannot be used with test, invokes checksumming for all files.

-t Test runs scripts without actually doing backup.

-s Silent, runs without screen output (I guess without sound would be -b).

-noa No autoconfiguration. Disables the autoconfiguration for a specific job.

-ob Specifies the pathname of the $OBK_HOME directory.

`Registering a DB in EBU`. A database must be registered in the EBU catalog before it can be backed up using EBU; an example registration sequence would be

```
register
  db_name = "TEST1"
  oracle_home = "/bto/sys/oracle/product/7.2.3"
  oracle_sid = "ORTEST1"
  pfile = "/oracle0/ORTEST1/admin/pfile/initORTEST1.ora"
log = "/oracle0/ORTEST1/admin/log/obkTEST1.log"
```

Once a database is registered, it can be backed up using commands similar to the following.

For on-line:

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
backup online database
 db_name = "TEST1"
 oracle_sid = "ORTEST1"
 parallel = 3
```

For off-line:

```
backup offline database
 db_name = "TEST1"
 oracle_sid = "ORTEST1"
parallel = 4
```

For tablespace backup:

```
backup online
 db_name = "TEST1"
 oracle_sid = "ORTEST1"
tablespace = "APPL_DATA","APPL_INDEX"
```

For control file backup:

```
backup online
 db_name = "TEST1"
 oracle_sid = "ORTEST1"
control_file
```

Control file and tablespace backups can be done together, as can database and control file.
An example shell script for UNIX backup is included with the EBU facility in the directory

```
$ORACLE_HOME/obackup/admin/lightsout.sh.
```

Restoration of Databases Using EBU. The EBU facility also allows for restoration of the database either to its original locations or to another host. A restore can consist of one or more of the following items:

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- Data files
- Control file
- Parameter (initialization file)
- Archive redo log file

Restoration of entire database:

```
restore database consistent
 db_name = "TEST1"
 oracle_sid = "ORTEST1"
 parallel = 4
log = "/oracle0/ORTEST1/admin/log/obkORTEST1.log"
```

Restore of tablespaces and control file:

```
restore
 db_name = "TEST1"
 oracle_sid = "ORTEST1"
 control_file
tablespace = "APPL_DATA","APPL_INDEX"
```

Restore with datafile move:

```
restore
 db_name = "TEST1"
 oracle_sid = "ORTEST1"
 dbfile = "/oracle1/ORTEST1/data/appl_data01.dbf"
 remap = "/oracle1/ORTEST1/data/appl_data01.dbf" to
"/oracle5/ORTEST1/data/appl_data01.dbf"
```

You can also do a point-in-time restore:

```
restore database
 db_name = "TEST1"
 oracle_sid = "ORTEST1"
to = "01/07/1997 13:00"
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The EBU facility also provides utilities for maintaining the catalog and generating reports on backups. I suggest a complete review of `Oracle7 Enterprise Backup Utility Administrator's Guide`, Release 2.0, March 1996, Part No. A42580-2 (Oracle Corporation) before attempting to use the EBU facility. It contains important information that time constraints prevented me from covering in this lecture.

Oracle8 RECOVERY MANAGER FACILITY

Oracle8 introduces the recovery manager RMAN, which is the Enterprise Backup Utility on steroids. RMAN allows backup of database files at the block level and automatically performs datafile compression by only backing up blocks that have been used or altered. In incremental mode, the RMAN only backs up blocks that have been altered or added in the database, greatly reducing the size of required backups.

RMAN also allows the following:

- Scripting with RMAN script language, backup, and restore operations.
- Reports on backup status and backup file status.
- Use of a recovery catalog to facilitate backup and restore operations.
- Parallelization of backup and restore operations.
- Backup based on specified limits (i.e., amount of redo generated against a file).
- Backup of database, tablespace, or individual data files.
- Batch backup operations.

RMAN uses a recovery catalog; however, you can use RMAN without a catalog from just the data stored in the control files, but

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

you are restricted to a subset of RMANs capabilities in this mode. The catalog contains information on the following:

- Data file and archive log backup sets and pieces
- Data file copies
- Archived redo logs and copies of them
- Tablespaces and data files at the target database
- Named, user-created sequences of commands called stored scripts

It is a good practice to maintain a small database strictly for the recovery catalog and perhaps the Enterprise Manager catalog files. The catalog should be resynchronized with all remote databases on a periodic basis. If you don't use a catalog you cannot do the following:

- Point-in-time recover
- Use stored scripts
- Recovery if a control file is not available.

RMAN creates backup sets that consist of backup pieces. Backup pieces are parts of the backup set at a size that is predetermined and usually based on backup media capacity of operating system file size limitations. Backup sets can be written to disk or secondary storage, can include a backup control file, and can span multiple OS files (pieces). Backup devices that are supported on your system are cataloged in the v$backup_device dynamic performance table.

RMAN backup sets that contain archive logs are called, appropriately enough, archivelog backup sets. With Oracle8 you cannot write archive logs directly to tape, but a job can be

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

scheduled using RMAN to back archive log backup sets to tape or other storage.

RMAN produces either full or incremental backups. A full backup is a backup of one or more data files that contains all blocks of the data file(s) that have been modified or changed. Full backups can be created out of:

- Data files
- Data file copies
- Tablespaces (all data files for a tablespace)
- Archive logs
- Control files (current or backups)
- Entire databases

An incremental backup is a backup of one or more files and contains only blocks that have been modified. However, only complete control files are backed up in either incremental or full backups. Incremental backups can be made of:

- Data files
- Tablespaces
- Databases

The incremental backup allows leveling of backups. Each level is denoted by an integer value, with the level of backup meaning that any blocks changed since the last incremental backup at this level will be backed up the next time this level is specified. This allows levels to be set based on time frames; for example, 0 being a monthly full, 1 being a once-a-week incremental, and 2 being a daily incremental. Of course, this also leads to complicated rotation

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

of tapes or backup media, taking us back to the good old towers-of-Hanoi backup scenario nightmares.

RMAN also allows for image copies of data files, archive logs, or control files. Image copies can only be made to disk and cannot contain multiple files.

RMAN allows report generation. Reports can be generated based on:

- What files need backup.
- What files haven't been backed up recently.
- What backup files can be deleted.

Each backup set can be associated with a tag that can be used to identify it in subsequent operations. The tag doesn't have to be unique. RMAN selects the most recent backup set in the case of backup sets with duplicate tags.

RMAN works against running or shutdown databases whether they are in archive log mode or not. However, if the database is not in archive log mode the entire database can only be backed up if it was shut down cleanly. Tablespaces can only be backed up in NOARCHIVELOG mode if they are off-line normal. There are no restrictions of this type on databases in ARCHIVELOG mode.

RMAN automatically detects corruptions and logs these in v$backup_corruption and v$copy_corruption dynamic performance tables. Corrupt blocks are still backed up.

Installing the RMAN Catalog

The catalog should be owned by a user with the resource role grant. I suggest a user in a small database dedicated to system administration functions such as the RMAN catalog and Enterprise

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Manager catalog. Create a tablespace for use by the RMAN user and assign that as the user's default tablespace with unlimited quota. For example, if we wanted our user to be named rman_dba, the steps would be as follows:

```
sqlplus system/manager
SQL>CREATE TABLESPACE rman_data DATAFILE 'file_spec'
DEFAULT STORAGE (clause);
SQL>CREATE USER rman_dba IDENTIFIED BY rman_dba
 2: DEFAULT TABLESPACE rman_data
 3: TEMPORARY TABLESPACE  temp
 4: QUOTA UNLIMITED ON rman_data;
SQL>GRANT RESOURCE,CONNECT TO rman_dba;
SQL>CONNECT rman_dba/rman_dba
SQL> @$ORACLE_HOME/rdbms/admin/catrman.sql
```

Once the catalog is built, the recovery manager can be utilized. The command is either rman, rman80, or RMAN80 depending on your operating system. There are literally dozens of commands for use with the RMAN facility. I suggest reviewing the `Oracle8i Server Backup and Recovery Guide`, Release 8.1.5 (or most current release) (Oracle Corporation, 1999) before using RMAN.

The following are some example scenarios showing how the commands can be made into scripts.
Connection to rman in UNIX on early versions can be tricky. On some UNICES the double quote (") character has to be escaped, and you need to use the double quotes to log into rman (at least on early versions). Assuming the database to be backed up is ORTEST1 with a TNS alias of ORTEST1, the user is as specified earlier, and the catalog database is ORRMAN, the connection to RMAN for the user SYSTEM password MANAGER would look like this:

```
$ rman ORTEST1\ system/manager@ORTEST1 rcvcat
"rman_dbo/rman_dbo@ORRMAN\"
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Intuitive, isn't it? A sample session from Recovery Manager (RMAN) is shown next.

The target database service name in the "tnsnames.ora" file is "ORTEST1." The recovery catalog database service name in the "tnsnames.ora" file is "ORRMAN."

```
% cd $ORACLE_HOME/rdbms/admin
% sqlplus sys/change_on_install@ORRMAN
SQL> grant connect, resource to RMAN_DBA identified by
RMAN_DBA;
Grant succeeded.
SQL> connect rman/rman@ORRMAN
Connected.
SQL> @catrman.sql
SQL> exit
%
% rman 'target sys/change_on_install@ORTEST1 rcvcat
rman/rman@ORRMAN'
Recovery Manager: Release 8.0.2.0.0 - Beta
RMAN-06005: connected to target database: ORTEST1
RMAN-06008: connected to recovery catalog database
RMAN> register database;
RMAN-08006: database registered in recovery catalog
RMAN-08002: starting full resync of recovery catalog
RMAN-08004: full resync complete
RMAN> run
2> {
3> allocate channel c1 type disk;

4> backup full format
'/oracle16/ORTEST1/amin/backup/backup%s%p' (database);
5> }

RMAN-08030: allocated channel: c1

RMAN-08500: channel c1: sid=12 devtype=DISK

RMAN-08008: channel c1: started datafile backupset
RMAN-08502: set_count=9 set_stamp=280246639

RMAN-08011: channel c1: including current controlfile in
backupset
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
RMAN-08010: channel c1: including datafile number 1 in
backupset
RMAN-08010: channel c1: including datafile number 2 in
backupset
     .
     .
     .
RMAN-08010: channel c1: including datafile number 11 in
backupset
RMAN-08010: channel c1: including datafile number 12 in
backupset

RMAN-08013: channel c1: piece 1 created

RMAN-08503: piece
handle=/oracle16/ORTEST1/admin/backup/backup91 comment=NONE
RMAN-08003: starting partial resync of recovery catalog
RMAN-08005: partial resync complete
RMAN-10030: RPC call appears to have failed to start on
channel default
RMAN-10036: RPC call ok on channel default
RMAN-08031: released channel: c1
RMAN> exit
```

Incomplete restore scenario

The following shows the scenario for an incomplete recovery. The following scenario assumes that:

- You wish to do an incomplete recovery due to an application error that was made at a specific time.
- There are three tape drives.
- You are using a recovery catalog.

TIP:

It is highly advisable to back up the database immediately after opening the database resetlogs.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The following script restores and recovers the database to the time immediately before the user error occurred. The script does the following:

- Starts the database mount and restricts connections to DBA-only users.
- Restores the database files (to the original locations).
- Recovers the data files by either using a combination of incremental backups and redo or just redo. Recovery Manager will complete the recovery when it reaches the transaction from the time specified.
- Opens the database resetlogs.

Oracle recommends that you backup your database after the resetlogs (this is not shown in the example).

Ensure that you set your NLS_LANG and NLS_DATE_FORMAT environment variables. You can set these to whatever you wish—the date format of the following example is the standard date format used for recovery, e.g., for UNIX (csh):

```
> setenv NLS_LANG AMERICAN
> setenv NLS_DATE_FORMAT 'YYYY-MM-DD:hh24:mi:ss'
```

Next, start up Server Manager:

```
SVRMGR> connect internal
Connected.
SVRMGR> startup mount restrict
SVRMGR>exit

#  rman target internal/knl@prod1 rcvcat rman/rman@rcat
cmdfile case2.rcv
run {
#  The 'set until time' command is for all commands
executed
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
#  between the { and } braces. Means both restore and
recover
#  will both be relative to that point in time.
#  Note that Recovery Manager uses the Recovery Catalog to,
#  determine the structure of the database at that time,
and
#  restore it.
#
    set until time '1997-06-23:15:45:00';
#
    allocate channel t1 type 'SBT_TAPE';
    allocate channel t2 type 'SBT_TAPE';
    allocate channel t3 type 'SBT_TAPE';
#
    restore
      (database);
#
#  There is no need to manually catalog logs before
recovery,
#  as Recovery Manager does catalog resync from the current
#  control file.
#
    recover
      database;
#
    sql 'alter database open resetlogs';
```

The preceding scenarios are just examples of how to use the recovery manager. Please consult your manual before attempting to use the facility for production work. The RMAN readme file contains valuable insights into RMAN use and has several additional scenarios.

DB_VERIFY UTILITY

In the final section of this lesson I want to cover the DB_VERIFY utility. The DB_VERIFY utility is an external command line-based utility that is used to perform a physical structure integrity check on an off-line (shutdown) database. The utility can be used against backup files and on-line files or pieces of on-line files. The utility is used to be sure a backup database or data file is valid before

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

recovery. The utility can also serve as a diagnostic aid when corruption is suspected. Since it runs against a shutdown database it can perform checks significantly faster than export or other utilities. The utility is named differently on different platforms; for example, it may be called dbv (on SUN/Sequent) or something else on your system. Verify its name with the system-specific documentation you should have received (if you didn't, call your Oracle rep and complain). The utility only verifies cache-managed blocks.

The DB_VERIFY utility has the following general syntax:

dbf|dbverify|dbverf80 keyword=value,keyword=value…

Where the keywords are shown in table 1.

Keyword	Description	(Default)
FILE	File to Verify	(NONE)
START	Start Block	(First Block of File)
END	End Block	(Last Block of File)
BLOCKSIZE	Logical Block Size	(2048)
LOGFILE	Output Log	(NONE)
FEEDBACK	Display Progress	(0)

Table 1: DBVERIFY Commands

The following shows some example runs of the DB_VERIFY against an Oracle 7.3 database:

The following example shows how to get on-line help:

```
D:\Oracle2\Ortest1\Data>dbv help=y

DBVERIFY: Release 8.1.5.0.0 - Production on Fri Feb 4
11:59:26 2000

(c) Copyright 1999 Oracle Corporation.  All rights
reserved.
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
Keyword   Description        (Default)
-------------------------------------------
FILE      File to Verify     (NONE)
START     Start Block        (First Block of File)
END       End Block          (Last Block of File)
BLOCKSIZE Logical Block Size (2048)
LOGFILE   Output Log         (NONE)
FEEDBACK  Display Progress   (0)
```

This is sample output of verification for the file, sys1ortest1.dbf. The feedback parameter has been given the value 1000 to display one dot on screen for every 1000 blocks processed:

```
D:\Oracle2\Ortest1\Data>dbv file=sys1ortest1.dbf
blocksize=8192 feedback=1000

DBVERIFY: Release 8.1.5.0.0 - Production on Fri Feb 4
11:47:13 2000

(c) Copyright 1999 Oracle Corporation.  All rights
reserved.

DBVERIFY - Verification starting : FILE = sys1ortest1.dbf
.............

DBVERIFY - Verification complete

Total Pages Examined         : 12800
Total Pages Processed (Data) : 3419
Total Pages Failing   (Data) : 0
Total Pages Processed (Index): 1304
Total Pages Failing   (Index): 0
Total Pages Processed (Other): 527
Total Pages Empty            : 7550
Total Pages Marked Corrupt   : 0
Total Pages Influx           : 0
```

The DBMS_REPAIR Utility

New in Oracle8i is the DBMS_REPAIR utility. The DBMS_REPAIR utility consists of a stored package of procedures and functions that

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

allow the DBA to detect and repair corrupt blocks in tables and indexes. This functionality has been sorely needed in Oracle for a long time. Now, instead of having to go through a complex recovery procedure should a single block become corrupted the DBA has the option of attempting repair using the DBMS_REPAIR utility.

DBMS_REPAIR Enumeration Types

In Oracle you can define constants in a package that can then be used throughout the package and database. Oracle has recently taken to calling these constants "enumeration types". Essentially an enumeration type is a global variable which defines to a constant numeric value. Enumeration types are used to assign values to specific function and procedure IN type variables. The DBMS_REPAIR package has the following enumeration types as shown in table 2.

Type	Values
Object_type	TABLE_OBJECT, INDEX_OBJECT, CLUSTER_OBJECT
Action	CREATE_ACTION, DROP_ACTION, PURGE_ACTION
Table_type	REPAIR_TABLE, ORPHAN_TABLE
Flags	SKIP_FLAG, NOSKIP_FLAG

Table 2: Enumeration Types For DBMS_REPAIR Package

The default table_name is REPAIR_TABLE when table_type is REPAIR_TABLE and will be ORPHAN_KEY_TABLE when table_type is ORPHAN.

DBMS_REPAIR Exceptions

The DBMS_REPAIR package can raise several self-declared exceptions. The DBMS_REPAIR self-declared exceptions are all in the 24000 number range and are shown in table 3.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Exception	Purpose
24120	This is raised by the package procedures if an invalid parameter is passed.
24122	This is raised by the package if an incorrect block range is specified in BLOCK_START or BLOCK_END.
24123	An unimplemented feature was called.
24124	An invalid action was specified.
24125	The target object has been dropped or truncated since DBMS_REPAIR.CHECK_OBJECT was last run.
24127	The TABLESPACE parameter was specified with an action other than CREATE_ACTION.
24128	A partition name was specified for an object that isn't partitioned.
24129	A table name parameter was passed without the appropriate prefix.
24130	The orphan table specified for the repair doesn't exist.
24131	The orphan table specified doesn't have a proper definition.
24132	Table names are limited to 30 characters, the name specified is longer than this.

Table 3: Exceptions For the DBMS_REPAIR Package

DBMS_REPAIR Procedures

The DBMS_REPAIR package contains six procedures (as of this writing) that are usable callable and no user callable functions. The procedures are listed in table 4 and the details of their use delineated in the sections that follow.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Procedure Name	Description
ADMIN_TABLES	Allows for the administrative functions for the DBMS_REPAIR repair and orphan key tables such as create, purge and drop.
CHECK_OBJECT	Used to detect and report corruptions in tables or indexes.
DUMP_ORPHAN_KEYS	Used to report on index entries that point to rows in corrupt table blocks.
FIX_CORRUPT_BLOCKS	Marks blocks as software corrupt that have been flagged as corrupt by CHECK_OBJECT.
REBUILD_FREELISTS	Rebuilds and objects freelists.
SKIP_CORRUPT_BLOCKS	For tables and indexes with corrupt blocks, tells Oracle to either ignore the blocks during scans or to raise the ORA-01578 error.

Table 4: Procedures in DBMS_REPAIR

The following sections describe each procedure and give an example of its use.

ADMIN_TABLES

The ADMIN_TABLES procedure has the following input variables and no output variables as shown in table 5:

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Argument	Type	In/Out	Default Value
TABLE_NAME	VARCHAR2	IN	GENERATE_DEFAULT_TABLE_NAME
TABLE_TYPE	BINARY_INTEGER	IN	
ACTION	BINARY_INTEGER	IN	
TABLESPACE	VARCHAR2	IN	NULL

Table 5: ADMIN_TABLES Input Variables

The procedure is used to create, purge and drop the REPAIR_TABLE and ORPHAN_KEY_TABLE which are used during the repair of database tables and indexes. If the TABLE_TYPE is set to REPAIR_TABLE then the GENERATE_DEFAULT_TABLE_NAME setting tells the procedure to set the table name to REPAIR_TABLE if the TABLE_TYPE is set to ORPHAN_TABLE then the TABLE_NAME is set to ORPHAN_KEY_TABLE by the procedure if the GENERATE_DEFAULT_TABLE_NAME value is entered. This procedure is a good example why defaulted values should be placed at the end of your argument list. Since the TABLE_NAME attribute is first it means that to use the default for it you must specify the positional naming for all other parameters in order to use the default value, take a look at Listing 7 to see what I mean.

Listing 7 Example Use of DBMS_REPAIR.ADMIN_TABLES procedure.

First, let's build a repair table. Note that the name specified for the table is upper case, this is required, if you specify the name in lower case you will get an error on exception 24129.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
SQL> execute dbms_repair.admin_tables(
'REPAIR_TABLE',dbms_repair.repair_table,dbms_repair.create_
action);

PL/SQL procedure successfully completed.

SQL> desc repair_table
 Name
Null?    Type
 ----------------------------------------------------- ----
---- --------------
 OBJECT_ID                                             NOT
NULL NUMBER
 TABLESPACE_ID                                         NOT
NULL NUMBER
 RELATIVE_FILE_ID                                      NOT
NULL NUMBER
 BLOCK_ID                                              NOT
NULL NUMBER
 CORRUPT_TYPE                                          NOT
NULL NUMBER
 SCHEMA_NAME                                           NOT
NULL VARCHAR2(30)
 OBJECT_NAME                                           NOT
NULL VARCHAR2(30)
 BASEOBJECT_NAME
VARCHAR2(30)
 PARTITION_NAME
VARCHAR2(30)
 CORRUPT_DESCRIPTION
VARCHAR2(2000)
 REPAIR_DESCRIPTION
VARCHAR2(200)
 MARKED_CORRUPT                                        NOT
NULL VARCHAR2(10)
 CHECK_TIMESTAMP                                       NOT
NULL DATE
 FIX_TIMESTAMP
DATE
 REFORMAT_TIMESTAMP
DATE
```

Now let's create an orphan key table. The same admonishment about use of upper case applies.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
SQL> execute
dbms_repair.admin_tables('ORPHAN_KEY_TABLE',dbms_repair.orp
han_table,dbms_repair.create_action);

PL/SQL procedure successfully completed.

SQL> desc orphan_key_table
 Name                           Null?    Type
 ------------------------------ -------- --------------------------------
 SCHEMA_NAME                    NOT NULL VARCHAR2(30)
 INDEX_NAME                     NOT NULL VARCHAR2(30)
 IPART_NAME                     VARCHAR2(30)
 INDEX_ID                       NOT NULL NUMBER
 TABLE_NAME                     NOT NULL VARCHAR2(30)
 PART_NAME                               VARCHAR2(30)
 TABLE_ID                       NOT NULL NUMBER
 KEYROWID                       NOT NULL ROWID
 KEY                            NOT NULL ROWID
 DUMP_TIMESTAMP                 NOT NULL DATE
```

Here is an example of using the 'GENERATE_DEFAULT_TABLE_NAME default value.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
SQL> execute
dbms_repair.admin_tables('GENERATE_DEFAULT_TABLE_NAME',dbms
_repair.orphan_table, dbms_repair.create_action);

PL/SQL procedure successfully completed.

SQL> desc orphan_key_table;
 Name                                              Null?    Type
 ------------------------------------------------- -------- ---------------
 SCHEMA_NAME                                       NOT NULL VARCHAR2(30)
 INDEX_NAME                                        NOT NULL VARCHAR2(30)
 IPART_NAME                                                 VARCHAR2(30)
 INDEX_ID                                          NOT NULL NUMBER
 TABLE_NAME                                        NOT NULL VARCHAR2(30)
 PART_NAME                                                  VARCHAR2(30)
 TABLE_ID                                          NOT NULL NUMBER
 KEYROWID                                          NOT NULL ROWID
 KEY                                               NOT NULL ROWID
 DUMP_TIMESTAMP                                    NOT NULL DATE
```

And here is an example using the default value, which requires the positional specification type procedure call. If the defaults would have been placed after the required fields, this could have been avoided.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
SQL> execute dbms_repair.admin_tables(
table_type=>dbms_repair.orphan_table,action=>dbms_repair.dr
op_action);
PL/SQL procedure successfully completed.

SQL> execute dbms_repair.admin_tables(
table_type=>dbms_repair.orphan_table,action=>dbms_repair.cr
eate_action);
PL/SQL procedure successfully completed.

SQL> desc orphan_key_table
 Name                                      Null?    Type
 ----------------------------------------- -------- ---------------
 SCHEMA_NAME                               NOT NULL VARCHAR2(30)
 INDEX_NAME                                NOT NULL VARCHAR2(30)
 IPART_NAME                                         VARCHAR2(30)
 INDEX_ID                                  NOT NULL NUMBER
 TABLE_NAME                                NOT NULL VARCHAR2(30)
 PART_NAME                                          VARCHAR2(30)
 TABLE_ID                                  NOT NULL NUMBER
 KEYROWID                                  NOT NULL ROWID
 KEY                                       NOT NULL ROWID
 DUMP_TIMESTAMP                            NOT NULL DATE
```

The other actions such as purge, are accessed the same way as is demonstrated in Listing 7 for the CREATE_ACTION.

CHECK_OBJECT

The CHECK_OBJECT procedure has up to nine possible input values and one output value. Again, due to placement of the arguments in order to use the default values you must use positional specifications for any calls to this procedure unless you specify values for all of the parameters. Table 6 shows the parameters for the CHECK_OBJECT procedure.

Argument	Type	In/Out	Default?
SCHEMA_NAME	VARCHAR2	IN	
OBJECT_NAME	VARCHAR2	IN	
PARTITION_NAME	VARCHAR2	IN	NULL
OBJECT_TYPE	BINARY_INTEGER	IN	TABLE_ OBJECT

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Argument	Type	In/Out	Default?
REPAIR_TABLE_ NAME	VARCHAR2	IN	REPAIR_ TABLE
FLAGS	BINARY_INTEGER	IN	NULL
RELATIVE_FNO	BINARY_INTEGER	IN	NULL
BLOCK_START	BINARY_INTEGER	IN	NULL
BLOCK_END	BINARY_INTEGER	IN	NULL
CORRUPT_ COUNT	BINARY_INTEGER	OUT	

Table 6: CHECK_OBJECT Procedure IN and OUT Parameters
As you can see from doing a DESCRIBE on the procedure, the positioning of the CORRUPT_COUNT OUT variable after all of the values that have default values will force us to use positional nomenclature or specify values for all of the required input variables. Sometimes I wish the guys that write this stuff had to use it on a daily basis. AN example run of the procedure CHECK_OBJECT is shown in Listing 8. Another problem is that if you have more than 32,767 problem entries this procedure will fail on numeric overflow since the output is specified as a BINARY_INTEGER instead of a plain old NUMBER.

Listing 8 Example Execution of the CHECK_OBJECT Procedure

Can we get by without specifying the out value?

```
SQL> execute
dbms_repair.check_object('GRAPHICS_DBA','INTERNAL_GRAPHICS'
);
BEGIN
dbms_repair.check_object('GRAPHICS_DBA','INTERNAL_GRAPHICS'
); END;

      *
ERROR at line 1:
ORA-06550: line 1, column 7:
PLS-00306: wrong number or types of arguments in call to
'CHECK_OBJECT'
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
ORA-06550: line 1, column 7:
PL/SQL: Statement ignored
```

Nope, I guess not, let's put a variable and see what happens.

```
SQL> var x number;
SQL> execute
dbms_repair.check_object('GRAPHICS_DBA','INTERNAL_GRAPHICS'
,:x);
BEGIN
dbms_repair.check_object('GRAPHICS_DBA','INTERNAL_GRAPHICS'
,:x); END;

      *
ERROR at line 1:
ORA-06550: line 1, column 7:
PLS-00306: wrong number or types of arguments in call to
'CHECK_OBJECT'
ORA-06550: line 1, column 7:
PL/SQL: Statement ignored
```

Nope, Oracle is too smart for us. Let's use positional notation once again due to the inconsideration of the developer:

```
SQL> execute
dbms_repair.check_object(schema_name=>'GRAPHICS_DBA',
object_name=>'INTERNAL_GRAPHICS', corrupt_count=>:x);

PL/SQL procedure successfully completed.

SQL> print x

        X
---------
        0
```

So all that work just to find out we don't have a problem…oh well, I guess it is better not to have a problem.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

DUMP_ORPHAN_KEYS

If the search with CHECK_TABLE turns up corrupt blocks, the DUMP_ORPHAN_KEYS procedure is used to retrieve key values from the table or index to facilitate the rebuild of the damaged block. Again, notice in table 7 how the positioning of the KEY_COUNT OUT attribute forces us to use positional nomenclature to use this procedure. The DUMP_ORPHAN_KEYS procedure has seven possible input variables where only the first two are required and one required output variable. Another problem is that if you have more than 32,767 problem entries this procedure will fail on numeric overflow since the output is specified as a BINARY_INTEGER instead of a plain old NUMBER.

ArgumentName	Type	In/Out	Default Value?
SCHEMA_NAME	VARCHAR2	IN	
OBJECT_NAME	VARCHAR2	IN	
PARTITION_NAME	VARCHAR2	IN	NULL
OBJECT_TYPE	BINARY_INTEGER	IN	INDEX_ OBJECT
REPAIR_TABLE_ NAME	VARCHAR2	IN	'REPAIR_ TABLE'
ORPHAN_TABLE_ NAME	VARCHAR2	IN	'ORPHAN_ KEY_ TABLE'
FLAGS	BINARY_INTEGER	IN	NULL
KEY_COUNT	BINARY_INTEGER	OUT	

Table 7: DUMP_ORPHAN_KEYS Procedure IN and OUT Parameters

An example run of this procedure is shown in Listing 9.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
SQL> execute dbms_repair.dump_orphan_keys(
schema_name=>'GRAPHICS_DBA',object_name=>'PK_INTERNAL_GRAPH
ICS', key_count=>:x);

PL/SQL procedure successfully completed.

SQL> print x

         X
---------
         0
```

Listing 9 Example Run of the DUMP_ORPHAN_KEYS Procedure

Sorry I can't provide more exciting examples but I don't know of an easy way to generate corrupt blocks so I could demonstrate actual results.

FIX_CORRUPT_BLOCKS

The FIX_CORRUPT_BLOCKS procedure allows you to tell oracle to mark the blocks as software corrupt and thus skip them, or, leave them as is and generate errors. The procedure has six possible input variables two of which are required and one out parameter as shown in Table 8. Guess what, yep, the OUT parameter FIX_COUNT is once again placed so that we have to use positional nomenclature or specify values for all of the input parameters in order to use the procedure. Also, once again, the OUT parameter FIX_COUNT is a BINARY_INTEGER limiting the number of possible fixes to 32,767.

Argument	Type	In/Out	Default Value?
SCHEMA_NAME	VARCHAR2	IN	
OBJECT_NAME	VARCHAR2	IN	
PARTITION_NAME	VARCHAR2	IN	NULL
OBJECT_TYPE	BINARY_INTEGER	IN	TABLE_ OBJECT

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Argument	Type	In/Out	Default Value?
REPAIR_TABLE_ NAME	VARCHAR2	IN	'REPAIR_ TABLE'
FLAGS	BINARY_INTEGER	IN	NULL
FIX_COUNT	BINARY_INTEGER	OUT	

Table 8: FIX_CORRUPT_BLOCKS Procedure IN and OUT Parameters

An example execution of this procedure is shown in Listing 10.

Listing 10 Example Execution of the FIX_CORRUPT_BLOCKS Procedure

```
SQL> execute
dbms_repair.fix_corrupt_blocks(schema_name=>'GRAPHICS_DBA',
object_name=>'INTERNAL_GRAPHICS', fix_count=>:x);

PL/SQL procedure successfully completed.

SQL> print x

         X
---------
         0
```

REBUILD_FREELISTS

The procedure REBUILD_FREELISTS is used to rebuild the freelists of tables that have been repaired to reflect the loss of the corrupt blocks. This procedure has four possible inputs, two of which have default values as shown in table 9. Amazingly enough Oracle put the defaults at the end of the variable list so positional naming isn't required, however, the developer who created all of these procedures has obviously never heard of the UPPER function so you must specify your arguments in upper case or an error will occur.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Argument	Type	In/Out	Default?
SCHEMA_NAME	VARCHAR2	IN	
OBJECT_NAME	VARCHAR2	IN	
PARTITION_NAME	VARCHAR2	IN	NULL
OBJECT_TYPE	BINARY_INTEGER	IN	TABLE_ OBJECT

Table 9: REBUILD_FREELISTS Procedure IN Parameters

An example run of this procedure is shown in Listing 11.

Listing 11 Example Run of the REBUILD_FREELISTS Procedure

Let's try lower case and see how user friendly this procedure is:

```
SQL> execute
dbms_repair.rebuild_freelists('graphics_dba','internal_grap
hics');
BEGIN
dbms_repair.rebuild_freelists('graphics_dba','internal_grap
hics'); END;
*
ERROR at line 1:
ORA-00942: table or view does not exist
ORA-06512: at "SYS.DBMS_REPAIR", line 278
ORA-06512: at line 1
```

I guess the answer is not very. Let's do it again with upper case inputs:

```
SQL> execute
dbms_repair.rebuild_freelists('GRAPHICS_DBA','INTERNAL_GRAP
HICS');

PL/SQL procedure successfully completed.
```

If there is only one freelist group, the master freelist is updated with all free blocks and the other freelists are zeroed. If the object has

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

multiple freelist groups then the master freelist in each freelist group is updated in a round-robin fashion and the rest of the freelists are zeroed. One question, since this procedure will be required to be executed after any run of the FIX_CORRUPT_BLOCKS procedure, why wasn't the functionality simply added to that procedure?

SKIP_CORRUPT_BLOCKS

The final procedure in the DBMS_REPAIR package is the SKIP_CORRUPT_BLOCKS procedure. The SKIP_CORRUPT_BLOCKS procedure is used to mark the corrupt blocks software corrupt and tell Oracle to skip those blocks during table and index scans. If the object specified is a cluster, it applies to all of the tables in the cluster and their respective indexes. The SKIP_CORRUPT_BLOCKS procedure has four possible inputs, two of which have default values as shown in Table 10. Amazingly enough Oracle put the defaults at the end of the variable list so positional naming isn't required, however, the developer who created all of these procedures has obviously never heard of the UPPER function so you must specify your arguments in upper case or an error will occur.

Argument	Type	In/ Out	Default Value?
SCHEMA_NAME	VARCHAR2	IN	
OBJECT_NAME	VARCHAR2	IN	
OBJECT_TYPE	BINARY_INTEGER	IN	TABLE_OBJECT
FLAGS	BINARY_INTEGER	IN	SKIP_FLAG

Table 10: SKIP_CORRUPT_BLOCKS Procedure IN Parameters

An example run of the SKIP_CORRUPT_BLOCKS procedure is shown in Listing 12.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
SQL> execute
dbms_repair.skip_corupt_blocks('GRAPHICS_DBA','INTERNAL_GRA
PHICS');

PL/SQL procedure successfully completed.
```

Listing 12 Example Run of the SKIP_CORRUPT_BLOCKS procedure

Oracle RDBMS Architecture

Oracle version 8i is an object-relational database management system (ORDBMS). Oracle8i (actually 8.1.5 production release) expands upon the new features offered in Oracle8 and makes over 150 changes or additions to the Oracle tools. A traditional RDBMS stores data in tables called `relations`. These relations are two-dimensional representations of data where the rows, called `tuples` in relational jargon, represent records, and the columns, called `attributes`, are the pieces of information contained in the record. Oracle8i provides new features in the object-oriented extensions provided in Oracle8 as well as to the Oracle RDBMS. In an object-relational database, columns can represent either a single value (as in standard relational databases), a varray (a fixed number of additional records), or a REF to a second table where a variable amount of data can be stored. This takes the two-dimensional relational view and adds a third dimension. In addition, in an object-relational database, procedures known as methods can be tied to the tables. Methods are above and beyond the old concept of triggers, as we shall see later. In latter releases of Oracle8i, JAVA a new object oriented language, can be used to create stored objects in an Oracle8i database.
Oracle consists of background processes, files, and executables. Lets look at these items.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Background Processes

Oracle is more than just a collection of programs that allow ease of data access. Oracle can be compared to an operating system that overlays the operating system of the computer on which it resides. Oracle has its own file structures, buffer structures, global areas, and tunability above and beyond those provided within the operating system. Oracle controls its own processes, controls its own records and consistencies, and cleans up after itself.

Oracle as it exists on your system (with the exception of DOS or OS/2) consists of executables, five to nine (or more) detached processes, a global memory area, data files, and maintenance files. It can be as small as a couple of megabytes, or as large as a massive globe-spanning construction of gigabytes. A diagram showing a typical Oracle8 and Oracle8i environment is shown in Figures 1 and 2; you may want to refer to these diagrams as you read the next sections.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

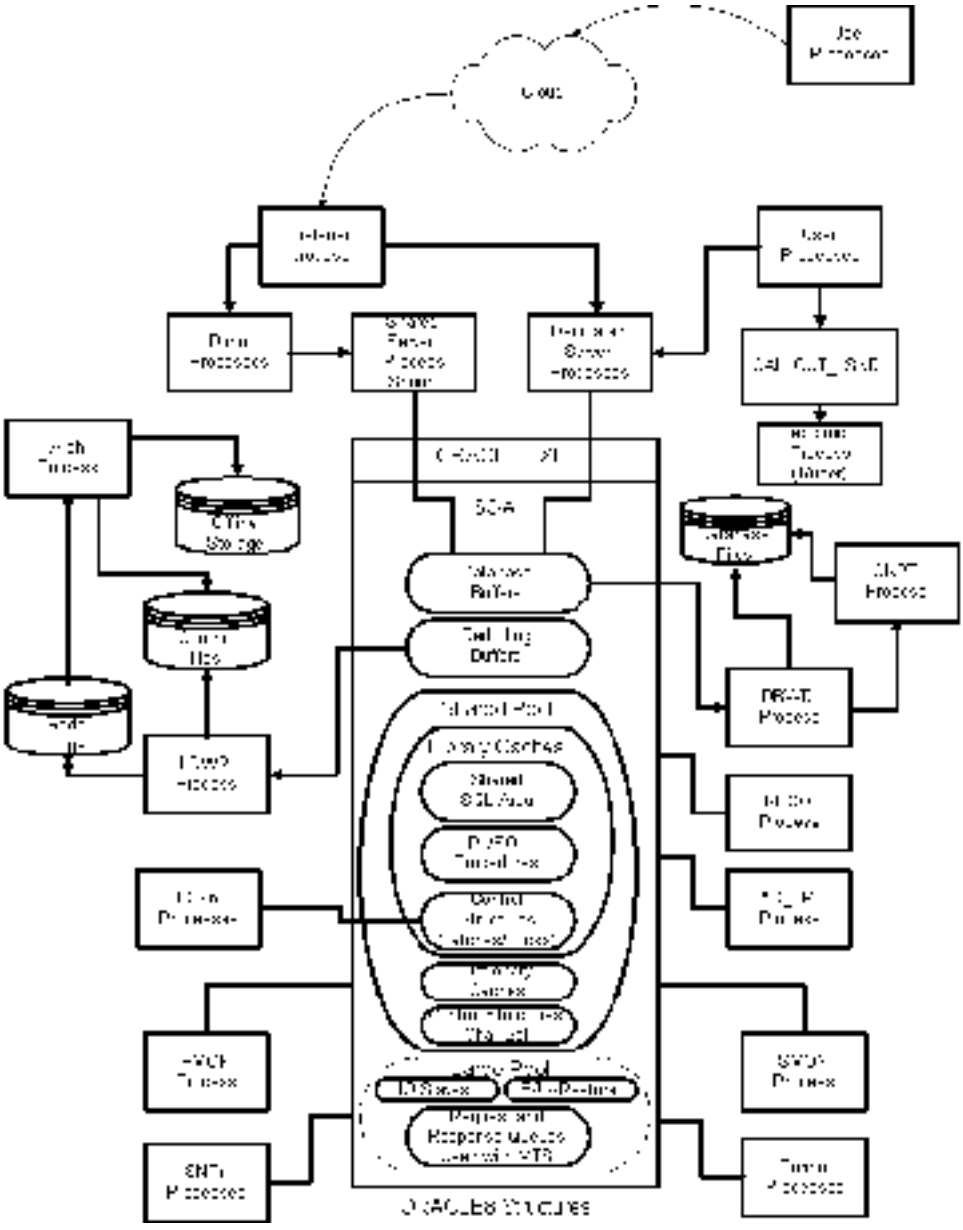


Figure 1: Oracle 8 Structures

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

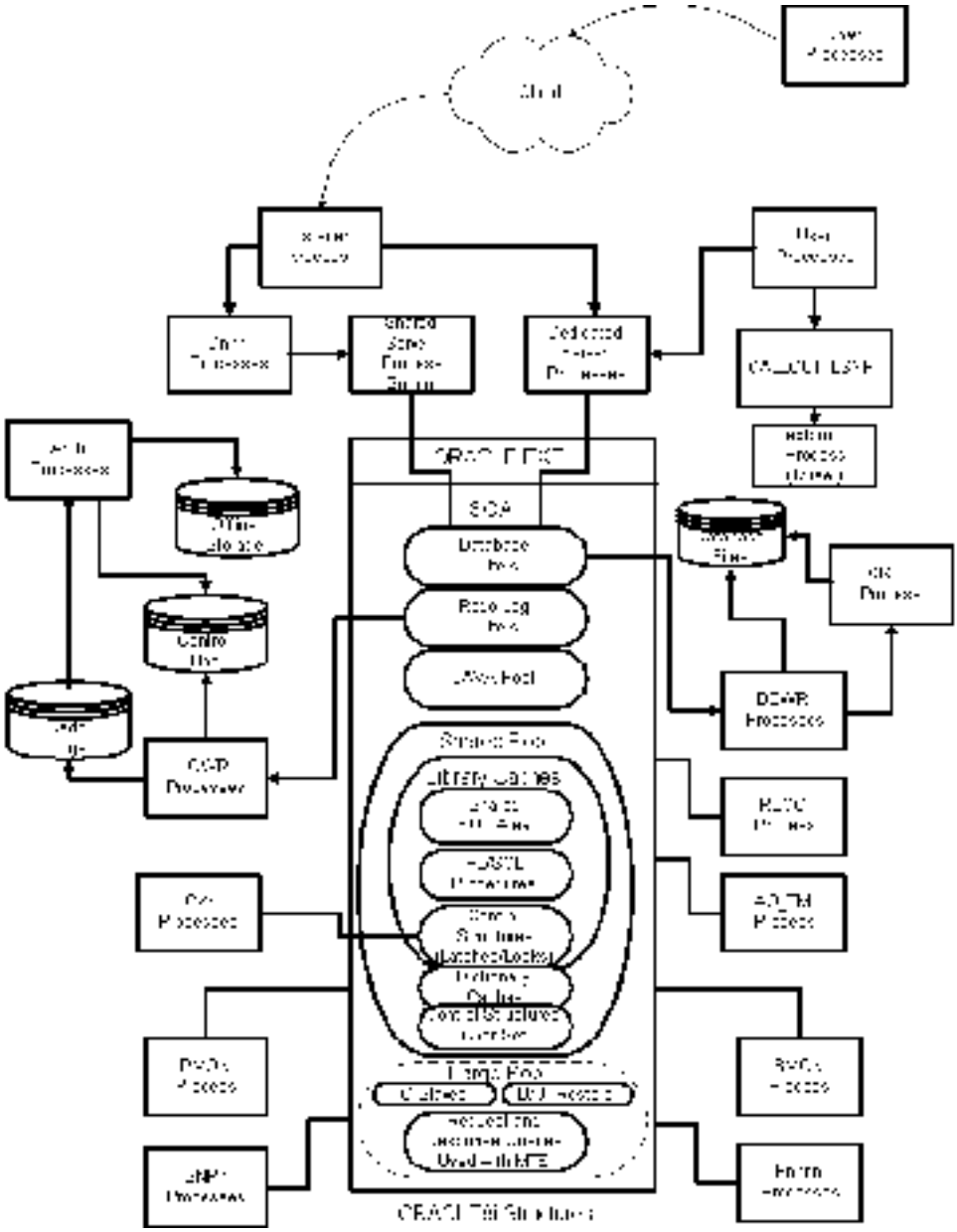


Figure 2: Oracle8i Structures

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

On VMS, NT (threads), or UNIX there may be a minimum of eight detached processes for Oracle7; for Oracle8i, this jumps to nearly a dozen. Four of these are the base Oracle processes and these processes are started every time Oracle is started up on a system; the additional processes may be started if the database is using archiving, uses TCPIP, or is being run in parallel and/or distributed mode. The Oracle job queues, snapshot processes, advanced queuing options, and callout processes all add to the process count. These processes are listed below.

- DBWR—Database Writer --This process handles data transfer from the buffers in the SGA to the database files.
- LGWR—Log Writer -- This process transfers data from the redo log buffers to the redo log database files.
- SMON—System Monitor -- This process performs instance recovery on instance startup and is responsible for cleaning up temporary segments. In a parallel environment, this process recovers failed nodes.
- PMON—Process Monitor -- This process recovers user processes that have failed and cleans up the cache. This process recovers the resources from a failed process.
- ARCH—Archiver Process -- This process is active only if archive logging is in effect. It writes the redo log data files that are filled into the archive log data files.
- RECO—Distributed Transaction -- This is an Oracle7 process that resolves failed distributed processes.
- LCK`n`—Lock Process -- This process is used for inter-instance locking in an Oracle7 parallel server environment.
- D`nnn`—Dispatcher -- This process allows multiple processes to share a finite number of Oracle7 servers. It queues and routes process requests to the next available server.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- Snnn—Servers -- This Oracle7 process makes all the required calls to the database to resolve a user's requests. It returns results to the Dnnn process that calls it.
- LISTENERTCPIP server -- If you are running TCPIP, this process, known as the listener process, will be running as well (only one per node).
- CKPxx -- This is the checkpoint process that can be started to optimize the checkpoint operation for Oracle logging.
- Snpxx -- These are snapshot process and job queues. Thee can be up to 32 configured in 8i.
- EXTPROC -- These are the callout queues; there will be one for each session performing callouts. It is hoped that Oracle will multithread these processes or it could result in the callout feature being unusable in a large multi-user environment.
- QMNn -- These are the Oracle queue monitor processes (new for 8i). They monitor the queues used in the Oracle advanced queuing (Oracle AQ). There can be up to 10 QMN processes. These take the place of the single AQ_TXX process in previous releases.

On multiuser-capable systems, each user process may spawn several sub-processes depending on the type of activities being done by that process. Depending on how Oracle is configured, a single parallel query may start dozens of query slave processes for a single user!

Datafiles

In Oracle datafiles are the physical implementation of the logical concept of tablespaces. Each tablespace contains segments which can be one of a specific set of segment types:

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- Data (both normal and partitioned)
- Index (both normal and partitioned)
- Rollback
- Temporary

Each segment is made up of extents and each extent is made from the finest level of granularity in the database, blocks.

Each datafile is allowed to sized either to 2, 4 or more gigabytes in size depending on operating system and patch level of Oracle.

Datafile placement has grown fuzzy in the last several years with the addition of RAID to the system administrators toolkit. It used to be before the wide spread use of RAID that a single disk platter could be assigned to handle a specific type of segment (and hence their datafiles). Now with striping, plaiding and other esoteric disk array arrangements it becomes harder and harder to determine what segment type is located on which disk.

As a DBA it will be your responsibility to:

1. Ensure datafiles are properly sized
2. Ensure, to the best of your ability, segment types are separated to ensure contention is realized
3. Ensure tablespaces are maintained to prevent fragmentation and poor space utilization
4. Ensure datafiles are properly backed up to allow for proper recovery in case of failure.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Datafile Sizing

Datafiles should be sized based on the size of the objects that will store segments in their related tablespaces. To this end, before tablespace and datafile sizing can be accomplished the size of tables and indexes contained in them must be determined. The Oracle DBA manual contains sizing procedures allowing a DBA to manually calculate sizing for both tables and indexes, however it is suggested that spread sheets similar to figures 3 and 4 be obtained or created to automate and standardize the calculations.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

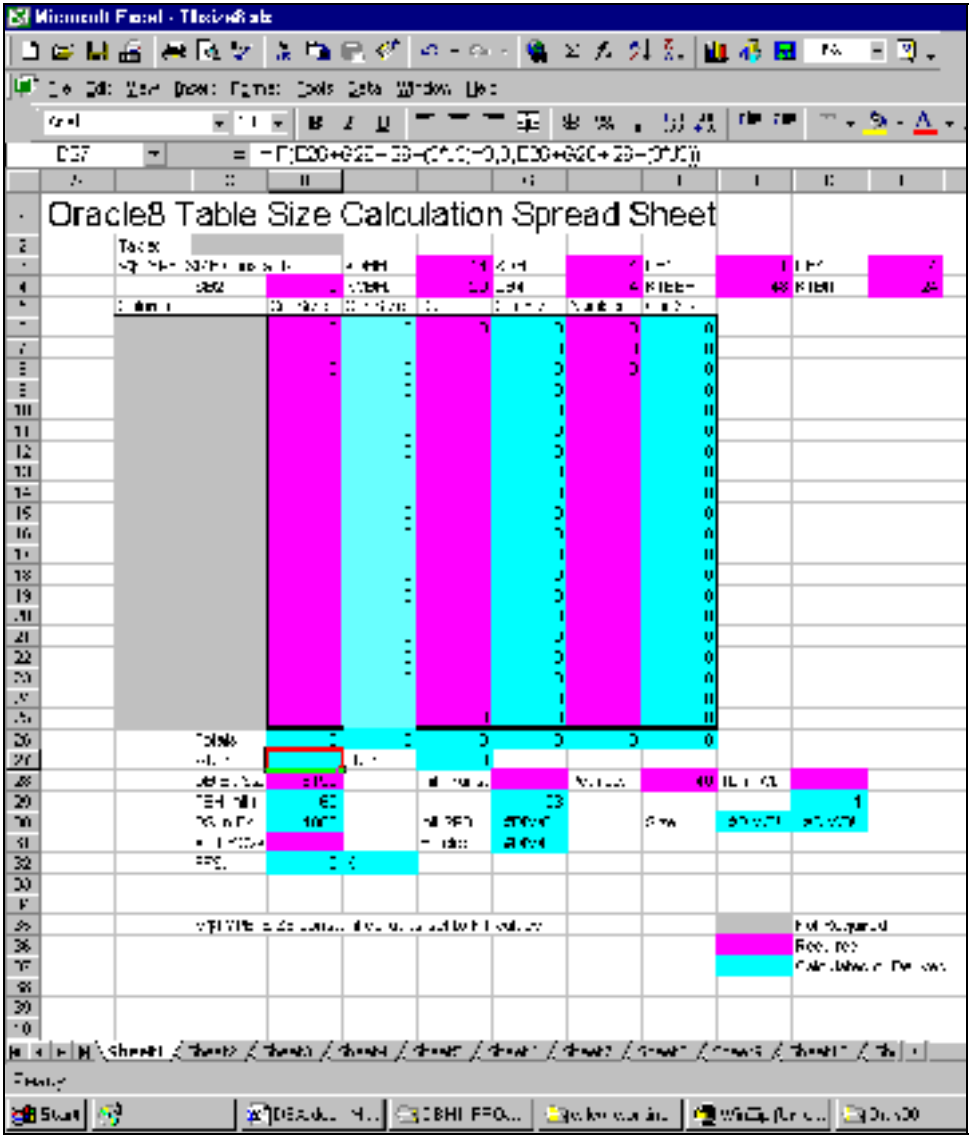


Figure 3: Example Table Sizing Spread Sheet

The table sizing spreadsheet shown in figure 3 only requires the input of the table column sizes in the appropriate column

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

(character, date and number) and the input of table specific items that deal with size such as initial transactions, percent free in a block, and number of rows. Additionally block size is specified. If the table is clustered then that too is taken into account to give a size estimate.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

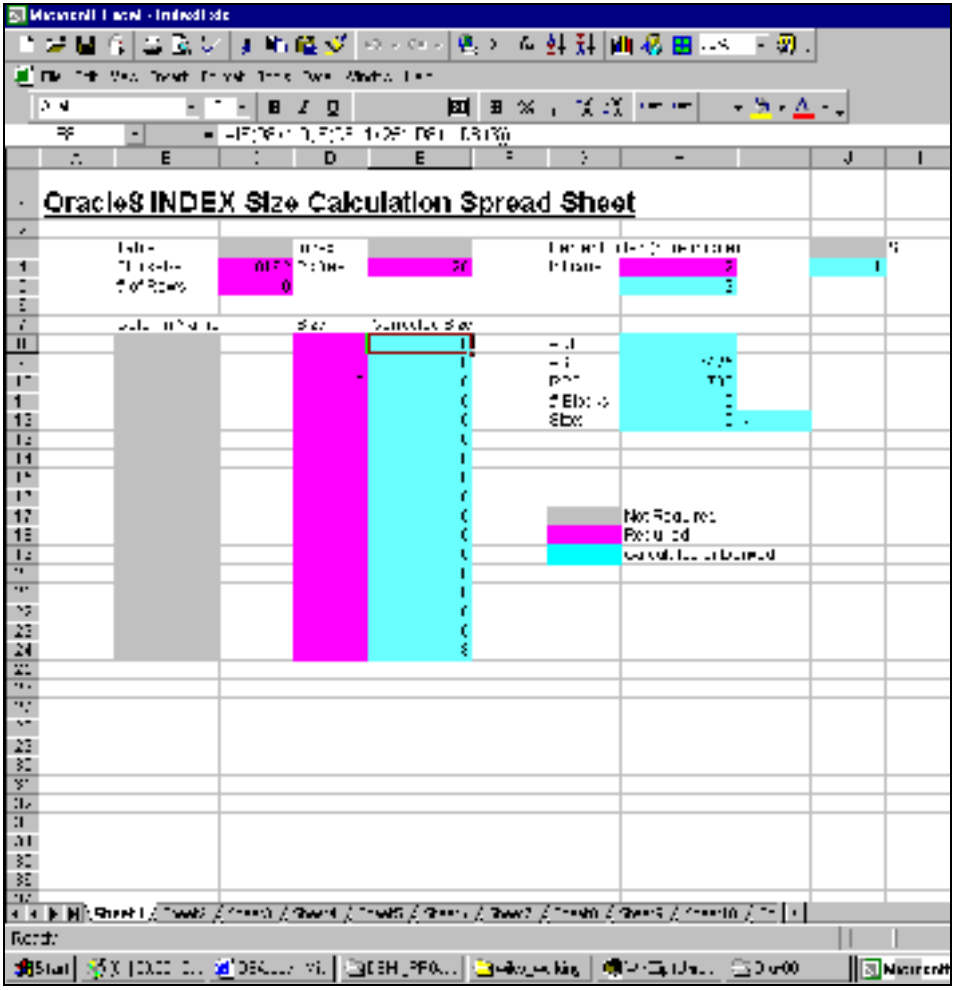


Figure 4: Example Index Sizing Spread Sheet

Using the index sizing spreadsheet shown above the only required inputs are the size of the columns to be included in the index, initial transactions, percent free, blocksize and number of rows expected. If the index is a non-unique index, you can also enter the expected percent of time the index column(s) will contain data.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Using standardized spreadsheets ensures that the calculation is performed identically and with repeatable results, it also allows for tweaking of the calculation until after the third or fourth time it is used it should be giving accurate estimates based on your system.

Once all of the tables and indexes are sized for the first year's data (or whatever time interval you desire) add the total sizes together for each object in each tablespace. These totals give the amount of space required for current data up to and including estimates for this year's growth, usually you double it to allow for an additional year. These adjusted totals become the sizes for the tablespaces. If the estimated sizes are too large for either operating system file size limits or available space, multiple datafiles will be required for that tablespace.

Also, by figuring out the total physical size for the database you can estimate the SGA base parameters such as DB_BLOCK_BUFFERS by use of a spreadsheet similar to the one shown in Figure 5.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

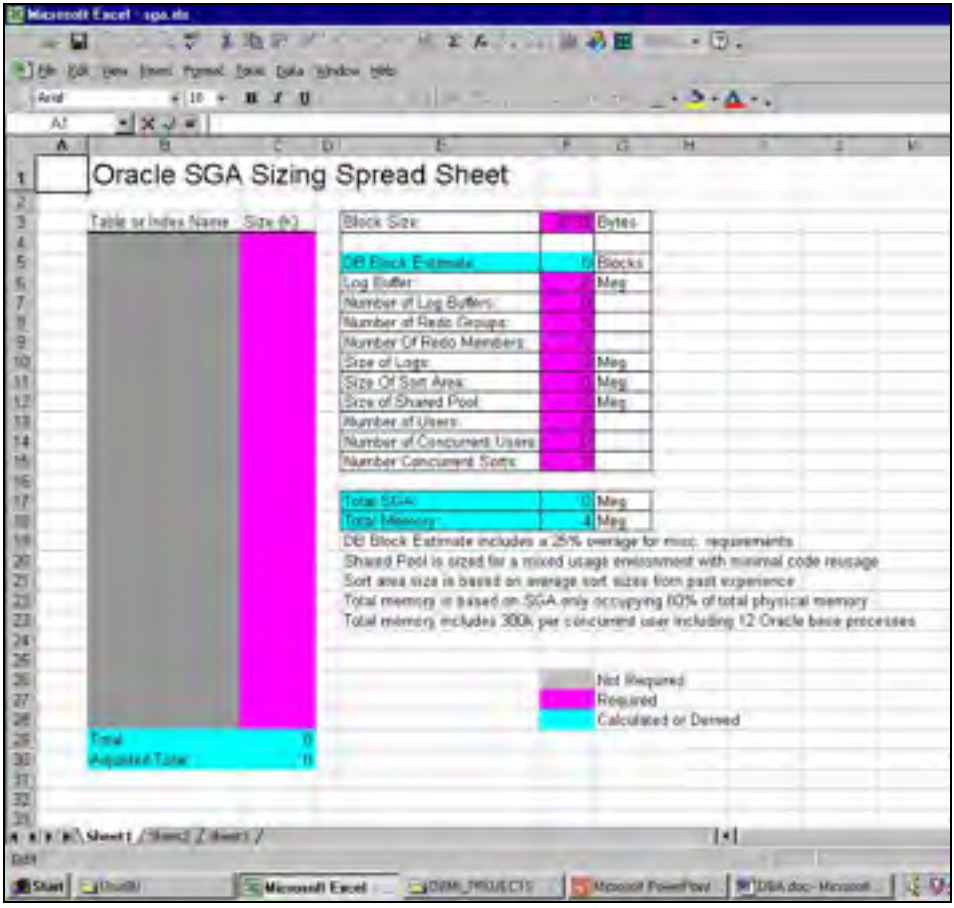


Figure 5: Example SGA Sizing Spreadsheet

Temporary tablespaces are used to hold user's sort segments when a sort operation exceeds the size of the memory allocated by the SORT_AREA_SIZE initialization parameter. Sorts to disk are on the average 14,000 times slower than memory sorts and so should be avoided if possible. Always assign a user a temporary tablespace, if one is not assigned the temporary tablespace defaults to the SYSTEM tablespace.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The sizing of the temporary tablespace is an empirical calculation involving assumed transaction sizes that involve sorts. Usually I set temporary tablespace storage at one megabyte initial and next and a percent increase of zero. I also set the initialization parameter SORT_AREA_SIZE to one megabyte and SORT_AREA_RETAINED_SIZE to one hundred kilobytes. Usually these values are enough, but if you have a sort intensive application you may have to increase the size of the sort area to reduce disk sorts. Any change in sort area size should also mean a change in the temporary tablespace initial and next extent sizes. The new views V$SORT_SEGMENT and V$SORT_USAGE should help in determining the proper settings for sort area size and the temporary tablespaces.

I suggest having multiple temporary tablespaces each serving a subset of users if your environment does heavy sorting, these multiple temporary tablespaces should be as physically separate as possible.

Rollback Segments

Rollback segments are used to store information on changed rows until those rows are rolled back or committed. If a rollback occurs, either from manual initiation using a ROLLBACK command or due to abnormal termination of a user process due to errors or failures, the data in the rollback segment is used to restore the database to the state before the transaction began.

The care and feeding or rollback segments is another empirical process. A few general observations:

1. If your users are doing heavy DML operations make sure there are enough properly sized rollback segment extents to allow each user performing DML to have a rollback segment extent without waiting.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

2. If you expect heavy DML loads, have multiple rollback segment tablespaces spread across multiple disk arrays.
3. Set the default sizing in a rollback segment tablespace such that you can simply issue a CREATE ROLLBACK SEGMENT command specifying the tablespace and you don't have to worry about the STORAGE specification.
4. Try to limit the number of simultaneous users per rollback segment to four. Size rollback segments such that:
    a. INITIAL=NEXT=size of average transaction
    b. OPTIMAL=size of average large transaction (rounded up to the next NEXT value)
    c. MINIMUM EXTENTS = OPTIMAL/INITIAL.
5. For large batch transactions create special large extent rollback segments in a separate tablespace from the other rollback segments, only bring them online when needed and use SET TRANSACTION command to assign them to specific transactions.
6. Try to avoid running large batch transactions simultaneously with OLTP or smaller transactions.

Generally you won't know the size of the average transaction, average large transaction or number of simultaneous DML users until the application has been running for a while. Use the DBA_ROLLBACK_SEGS, V$ROLLSTAT and V$ROLLNAME views to calculate the average values.

Rollback segments are assigned in a round-robin methodology. Each user gets assigned one extent to begin with which is used in a circular fashion if possible. When a user overwrites his own data it is called a WRAP. If the users data needs exceed the size of one rollback segment extent then a second is assigned, this is called an EXTEND. When a user forces a rollback segment to extend beyond the OPTIMAL setting for that rollback segment, the next user to use

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

the rollback segment must wait for the rollback segment to go back to the OPTIMAL setting, this is called a SHRINK.

When a WRAP, EXTEND or SHRINK occurs this results in an UNDO WAIT (either block or header) condition. A WRAP causes a very minor wait, an EXTEND into a newly assigned extent a slightly more expensive wait and the SHRINK the most expensive wait. By properly sizing rollback segment extents virtually all waits will be of the lower expense WRAP and EXTEND type and not the SHRINK type. Ideally all UNDO WAIT conditions should be caused by WRAP situations.

Redo log files

Redo logs are transaction journals. Each transaction is recorded in the redo logs. Redo logs are used in a serial fashion with each transaction queuing up in the redo log buffers and being written one at a time into the redo logs. Redo logs as a general rule should switch about every thirty minutes. However, you may need to adjust the time up or down depending on the importance of your data. The rule of thumb is to size the redo logs such that you only loose the amount of data you can stand to loose should for some reason the online redo log become corrupt. With modern Oracle redo log mirroring and with disk array mirroring and various forms of online disk repair and replacement the occurrence of redo log corruptions has dropped to practically zero, so size based on the number of archive logs you want to apply should the database fail just before your next backup.

The LOG_BUFFER_SIZE and LOG_BUFFERS parameters control the redo log buffers. The LOG_BUFFER_SIZE should be set to reduce the number of writes required per redo log but not be so large that it results in an excessive IO wait time. Some studies have shown that sizing bigger than one megabyte rarely results in performance gains. Generally I size the LOG_BUFFER_SIZE such that it is equal to or results in an even divisor of the redo log size.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Monitor redo logs using the alert log, V$LOGHIST, V$LOGFILE, V$RECOVERY_LOG and V$LOG DPTs.

Control files

Control files are created when the database is created. Control file size is determined by a number of initialization parameters and really shouldn't concern you. However, if you are using raw devices for control files make sure the raw partition is sized to allow the control file to grow in Oracle8 and Oracle8i databases as backup information is added. Each database instance must have at least one control file of it will not start. Loss of all control files will cause the database to crash. Have at least two copies on physically separate disk arrays or devices.

I suggest maintaining a script to rebuild your control files. The control file rebuild script should be recreated after each physical change to the database. The command CREATE CONTROLFILE is used to manually create a controlfile. However, the ALTER DATABASE BACKUP CONTROL FILE TO TRACE command is the preferred method to generate a script that can be used to rebuild the control file if it becomes damaged.

Initialization File

Although not normally discussed as a database file a database will not startup without a valid initialization parameter file. The file is only read at startup and contains the information required to set up the SGA, control file locations, trace and dump file locations and setup parameters for multi-threaded server, parallel query and Oracle Parallel Server. I suggest keeping a copy in a secure place incase yours ever becomes corrupted, deleted or damaged. The Oracle Reference, which is included in the Oracle8 and Oracle8i documentation set, provides descriptions of all initialization parameters. There are also undocumented initialization parameters that allow specialized behaviors to be configured. Both the

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

documented and undocumented initialization parameters are covered in my book “ORACLE8i Administration and Management” from Wiley and Sons.

It should be obvious that the most important file regarding database setup and operation is probably the INIT<sid>.ORA, or initialization file. This file contains the assignments for the database initialization parameters.

For Oracle7, version 7.3, there are 154 initialization parameters, for Oracle8, version 8.0.5, there are 184. In Oracle8i there are 194. Table 11 is a list of the Oracle8i INIT.ORA parameters, their default values, and descriptions. Note that on your platform there may be more or less than this number of parameters.

NAME	VALUE	DESCRIPTION
O7_DICTIONARY_ ACCESSIBILITY	TRUE	V7 Dictionary accessibility Support (Migration only)
ALWAYS_ANTI_JOIN	NESTED_ LOOPS	Always use this anti-join when possible
ALWAYS_SEMI_JOIN	standard	Always use this semi-join when possible
AQ_TM_PROCESSES	0	Number of AQ Time Managers to start
AUDIT_FILE_DEST	(PD)	Destination for audit files
AUDIT_TRAIL	NONE	Enable system auditing
BACKGROUND_CORE _DUMP	PARTIAL	Sets whether SGA is dumped with core file dump, PARTIAL means don't dump SGA.
BACKGROUND_DUMP _DEST	(PD)	Detached process dump directory
BACKUP_TAPE_IO_ SLAVES	FALSE	BACKUP Tape I/O slaves

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	VALUE	DESCRIPTION
BITMAP_MERGE_ AREA_SIZE	1048576	Maximum memory allow for BITMAP MERGE
BLANK_TRIMMING	FALSE	Blank trimming semantics parameter
BUFFER_POOL_KEEP	0	Number of database blocks/latches in KEEP buffer pool
BUFFER_POOL_ RECYCLE	0	Number of database blocks/latches in recycle buffer pool
COMMIT_POINT_ STRENGTH	1	Bias this node has toward not preparing in a two-phase commit
COMPATIBLE	8.1.0	Database will be compatible with this software version
CONTROL_FILE_ RECORD_KEEP_TIME	7	Control file record keep time in days
CONTROL_FILES	(PD)	Control file names list
CORE_DUMP_DEST	(PD)	Destination for core dump files.
CPU_COUNT	(PD)	Number of cpu's for this instance
CREATE_BITMAP_ AREA_SIZE	8388608	Size of create bitmap buffer for bitmap index
CURSOR_SPACE_ FOR_TIME	FALSE	Use more memory in order to get faster execution
DB_BLOCK_BUFFERS	8000	Number of database blocks cached in memory
DB_BLOCK_ CHECKING	TRUE	Data and index block checking overrides events 10210 and 10211

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	VALUE	DESCRIPTION
DB_BLOCK_ CHECKSUM	FALSE	Store checksum in db blocks and check during reads
DB_BLOCK_LRU_ LATCHES	1|CPU_CO UNT/2	Number of lru latches
DB_BLOCK_MAX_ DIRTY_TARGET	DB_BLOCK _BUFFERS	Upper bound on modified buffers/recovery reads
DB_BLOCK_SIZE	(PD)	Size of database block in bytes
DB_DOMAIN	WORLD	Directory part of global database name stored with CREATE DATABASE
DB_FILE_DIRECT_IO_ COUNT	64	Sequential I/O block count
DB_FILE_MULTIBLOCK _READ_COUNT	8	Db blocks read for each IO
DB_FILE_NAME_ CONVERT	NULL	Datafile name convert pattern and string for standby/clone database
DB_FILES	MAXDATAF ILES	Max allowable # db files
DB_NAME	(PD)	Database name specified in CREATE DATABASE
DB_WRITER_ PROCESSES	1	Number of background database writer processes to start
DBLINK_ENCRYPT_ LOGIN	FALSE	Enforce password for distributed login always be encrypted
DBWR_IO_SLAVES	0	Number of DBWR I/O slaves
DELAYED_LOGGING_ BLOCK_CLEANOUTS	TRUE	Turns delayed block cleanout on or off

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	VALUE	DESCRIPTION
DISK_ASYNCH_IO	TRUE	Use asynch I/O for random access devices
DISTRIBUTED_ TRANSACTIONS	(PD)	Max. number of concurrent distributed transactions
DML_LOCKS	4*Trans.	Dml locks - one for each table modified in a transaction
ENQUEUE_ RESOURCES	Derived	Resources for enqueues
EVENT	NULL	Debug event control
FIXED_DATE	NULL	Fixed SYSDATE value
FREEZE_DB_FOR_ FAST_INSTANCE_ RECOVERY	FALSE	Freeze database during instance recovery (OPS)
GC_DEFER_TIME	10	How long to defer down converts for hot buffers (DFS)(OPS)
GC_FILES_TO_LOCKS	NULL	Mapping between file numbers and lock buckets (DFS)(OPS)
GC_RELEASABLE_ LOCKS	0	Number of releasable locks (DFS)(OPS)
GC_ROLLBACK_ LOCKS	20	Locks for the rollback segments (DFS)(OPS)
GLOBAL_NAMES	TRUE	Enforce that database links have same name as remote database
HASH_AREA_SIZE	2*SORT	Size of in-memory hash work area
HASH_JOIN_ENABLED	TRUE	Enable/disable hash join
HASH_MULTIBLOCK_ IO_COUNT	1	Number of blocks hash join will read/write at once

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	VALUE	DESCRIPTION
HI_SHARED_MEMORY_ADDRESS	0	SGA starting address (high order 32-bits on 64-bit platforms)
HS_AUTOREGISTER	TRUE	Enable automatic server DD updates in HS agent self-registration
IFILE	NULL	Include file in init.ora
INSTANCE_GROUPS	NULL	List of instance group names
INSTANCE_NAME	NULL	Instance name supported by the instance
INSTANCE_NUMBER	0	Instance number
JAVA_POOL_SIZE	10000K	Size in bytes of the Java pool
JOB_QUEUE_INTERVAL	60	Wakeup interval in seconds for job queue processes
JOB_QUEUE_KEEP_CONNECTIONS	FALSE	Keep network connections between execution of jobs
JOB_QUEUE_PROCESSES	0	Number of job queue processes to start
LARGE_POOL_SIZE	0	Size in bytes of the large allocation pool (auto set at 600k)
LICENSE_MAX_SESSIONS	0	Maximum number of non-system user sessions allowed
LICENSE_MAX_USERS	0	Maximum number of named users that can be created in the database
LICENSE_SESSIONS_WARNING	0	Warning level for number of non-system user sessions

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	VALUE	DESCRIPTION
LM_LOCKS	12000	Number of locks configured for the lock manager (OPS)
LM_PROCS	64	Number of client processes configured for the lock manager (OPS)
LM_RESS	6000	Number of resources configured for the lock manager (OPS)
LOCAL_LISTENER	NULL	Local listener
LOCK_NAME_SPACE	NULL	Lock name space used for generating lock names for standby/clone database
LOCK_SGA	FALSE	Lock entire SGA in physical memory
LOG_ARCHIVE_DEST	(PD)	Archival destination text string
LOG_ARCHIVE_DEST_1	NULL	Archival destination #1 text string
LOG_ARCHIVE_DEST_2	NULL	Archival destination #2 text string
LOG_ARCHIVE_DEST_3	NULL	Archival destination #3 text string
LOG_ARCHIVE_DEST_4	NULL	Archival destination #4 text string
LOG_ARCHIVE_DEST_5	NULL	Archival destination #5 Text string
LOG_ARCHIVE_DEST_STATE_1	enable	Archival destination #1 state text string
LOG_ARCHIVE_DEST_STATE_2	enable	Archival destination #2 state text string
LOG_ARCHIVE_DEST_STATE_3	enable	Archival destination #3 state text string

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	VALUE	DESCRIPTION
LOG_ARCHIVE_DEST_STATE_4	enable	Archival destination #4 state text string
LOG_ARCHIVE_DEST_STATE_5	enable	Archival destination #5 state text string
LOG_ARCHIVE_DUPLEX_DEST	NULL	Duplex archival destination text string
LOG_ARCHIVE_FORMAT	ARC%S.%T	Archival destination Format. %s – log sequence %t - thread number
LOG_ARCHIVE_MAX_PROCESSES	1	Maximum number of active ARCH processes
LOG_ARCHIVE_MIN_SUCCEED_DEST	1	Minimum number of archive destinations that must succeed
LOG_ARCHIVE_START	TRUE	Start archival process on SGA initialization
LOG_BUFFER	500k or 128k*CPU	Redo circular buffer size
LOG_CHECKPOINT_INTERVAL	(PD)	# redo blocks checkpoint threshold
LOG_CHECKPOINT_TIMEOUT	0	Maximum time interval between checkpoints in seconds
LOG_CHECKPOINTS_TO_ALERT	FALSE	Log checkpoint begin/end to alert file
LOG_FILE_NAME_CONVERT	NULL	Logfile name convert pattern and string for standby/clone database
MAX_COMMIT_PROPAGATION_DELAY	700	Max age of new snapshot in .01 seconds
MAX_DUMP_FILE_SIZE	10000	Maximum size (blocks) of dump file

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	VALUE	DESCRIPTION
MAX_ENABLED_ ROLES	20	Max number of roles a user can have enabled
MAX_ROLLBACK_ SEGMENTS	30	Max. number of rollback segments in SGA cache
MTS_DISPATCHERS	NULL	Specifications of dispatchers
MTS_LISTENER_ ADDRESS	NULL	Address(es) of network listener
MTS_MAX_ DISPATCHERS	5	Max number of dispatchers
MTS_MAX_SERVERS	20	Max number of servers
MTS_MULTIPLE_ LISTENERS	FALSE	Are multiple listeners enabled?
MTS_SERVERS	1	Number of servers to start up
MTS_SERVICE	(DBD)	Service supported by dispatchers
MVIEW_REWRITE_ ENABLED	FALSE	Allow rewrite of queries using materialized views if enabled
NLS_CALENDAR	Gregorian	NLS calendar system name
NLS_COMP	BINARY	NLS comparison
NLS_CURRENCY	NULL	NLS local currency symbol
NLS_DATE_FORMAT	NULL	NLS Oracle date format
NLS_DATE_ LANGUAGE	NULL	NLS date language name
NLS_ISO_CURRENCY	NULL	NLS ISO currency territory name
NLS_LANGUAGE	AMERICAN	NLS language name
NLS_NUMERIC_ CHARACTERS	NULL	NLS numeric characters

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	VALUE	DESCRIPTION
NLS_SORT	NULL	NLS linguistic definition name
NLS_TERRITORY	AMERICA	NLS territory name
NLS_TIME_FORMAT	NULL	Time format
NLS_TIME_TZ_ FORMAT	NULL	Time with timezone format
NLS_TIMESTAMP_ FORMAT	NULL	Time stamp format
NLS_TIMESTAMP_TZ_ FORMAT	NULL	Timestamp with timezone format
NLS_UNION_ CURRENCY	NULL	Union currency symbol
OBJECT_CACHE_ MAX_SIZE_PERCENT	10	Percentage of maximum size over optimal of the user session's object cache
OBJECT_CACHE_ OPTIMAL_SIZE	102400	Optimal size of the user session's object cache in bytes
OPEN_CURSORS	64	Max # cursors per process
OPEN_LINKS	4	Max # open links per session
OPEN_LINKS_PER_ INSTANCE	4	Max # open links per instance
OPTIMIZER_ FEATURES_ENABLE	8.0.0	Optimizer plan compatibility parameter
OPTIMIZER_INDEX_ CACHING	0	Optimizer percent index caching
OPTIMIZER_INDEX_ COST_ADJ	100	Optimizer index cost adjustment
OPTIMIZER_MAX_ PERMUTATIONS	80000	Optimizer maximum join permutations per query block

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	VALUE	DESCRIPTION
OPTIMIZER_MODE	CHOOSE	Optimizer mode
OPTIMIZER_ PERCENT_PARALLEL	100	Optimizer percent parallel
OPTIMIZER_SEARCH_ LIMIT	5	Optimizer search limit
ORACLE_TRACE_ COLLECTION_NAME	NULL	Oracle TRACE default collection name
ORACLE_TRACE_ COLLECTION_PATH	(PD)	Oracle TRACE collection path
ORACLE_TRACE_ COLLECTION_SIZE	5242880	Oracle TRACE collection file max. size
ORACLE_TRACE_ ENABLE	FALSE	Oracle TRACE instance wide enable/disable
ORACLE_TRACE_ FACILITY_NAME	oracled	Oracle TRACE default facility name
ORACLE_TRACE_ FACILITY_PATH	(PD)	Oracle TRACE facility path
OS_AUTHENT_PREFIX	OPS$	Prefix for auto-logon accounts
OS_ROLES	FALSE	Retrieve roles from the operating system
PARALLEL_ ADAPTIVE_MULTI_ USER	FALSE	Enable adaptive setting of degree for multiple user streams
PARALLEL_ AUTOMATIC_TUNING	FALSE	Enable intelligent defaults for parallel execution parameters
PARALLEL_ BROADCAST_ ENABLED	FALSE	Enable broadcasting of small inputs to hash and sort merge joins
PARALLEL_ EXECUTION_ MESSAGE_SIZE	4096	Message buffer size for parallel execution
PARALLEL_ INSTANCE_GROUP	NULL	Instance group to use for all parallel operations

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	VALUE	DESCRIPTION
PARALLEL_MAX_ SERVERS	(PD)	Maximum parallel query servers per instance
PARALLEL_MIN_ PERCENT	0	Minimum percent of threads required for parallel query
PARALLEL_MIN_ SERVERS	0	Minimum parallel query servers per instance
PARALLEL_SERVER	FALSE	If TRUE startup in parallel server mode
PARALLEL_SERVER_ INSTANCES	1	Number of instances to use for sizing OPS SGA structures
PARALLEL_THREADS_ PER_CPU	2	Number of parallel execution threads per CPU
PARALLEL_ TRANSACTION_ RECOVERY	LOW	Max number of parallel recovery slaves that may be used
PARTITION_VIEW_ ENABLED	FALSE	Enable/disable partitioned views
PLSQL_V2_ COMPATIBILITY	FALSE	PL/SQL version 2.x compatibility flag
PRE_PAGE_SGA	FALSE	Pre-page sga for process
PROCESSES	100	User processes
READ_ONLY_OPEN_ DELAYED	FALSE	If TRUE delay opening of read only files until first access
RECOVERY_ PARALLELISM	0	Number of server processes to use for parallel recovery
REMOTE_ DEPENDENCIES_ MODE	TIMESTAM P	Remote-procedure-call dependencies mode parameter
REMOTE_LOGIN_ PASSWORDFILE	NONE	Password file usage parameter

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	VALUE	DESCRIPTION
REMOTE_OS_ AUTHENT	FALSE	Allow non-secure remote clients to use auto-logon accounts
REMOTE_OS_ROLES	FALSE	Allow non-secure remote clients to use os roles
REPLICATION_ DEPENDENCY_ TRACKING	TRUE	Tracking dependency for Replication parallel propagation
RESOURCE_LIMIT	FALSE	Master switch for resource limit
RESOURCE_MANAGE R_PLAN	NULL	Resource mgr top plan
REWRITE_INTEGRITY	enforce	Perform rewrite using materialized views with desired integrity
ROLLBACK_ SEGMENTS	(PD)	Private undo segment list
ROW_LOCKING	Always	Row-locking
SERIAL_REUSE	DISABLE	Reuse the frame segments
SERIALIZABLE	FALSE	Serializable
SERVICE_NAMES	(DBD)	Service names supported by the instance
SESSION_CACHED_ CURSORS	0	Number of cursors to save in the session cursor cache
SESSION_MAX_ OPEN_FILES	10	Maximum number of open bfiles allowed per session
SESSIONS	(1.1*PROC) +5	User and system sessions
SHADOW_CORE_ DUMP	PARTIAL	Include SGA in core dump (FULL) don't include it (PARTIAL)

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	VALUE	DESCRIPTION
SHARED_MEMORY_ ADDRESS	0	SGA starting address (low order 32-bits on 64-bit platforms)
SHARED_POOL_ RESERVED_SIZE	5% shared pool	Size in bytes of reserved area of shared pool
SHARED_POOL_SIZE	<64bit 16M	Size in bytes of shared pool
SNAPSHOT_ REFRESH_INTERVAL	60	Wakeup interval in seconds for job queue processes
SNAPSHOT_ REFRESH_KEEP_ CONNECTIONS	FALSE	Keep network connections between execution of refreshes
SNAPSHOT_ REFRESH_ PROCESSES	0	Number of job queue processes to start
SORT_AREA_ RETAINED_SIZE	sort_area_s ize	Size of in-memory sort work area retained between fetch calls
SORT_AREA_SIZE	(PD)	Size of in-memory sort work area
SORT_MULTIBLOCK_ READ_COUNT	2	Multi-block read count for sorts
SQL92_SECURITY	FALSE	Require select privilege for searched update/delete
SQL_TRACE	FALSE	Enable SQL trace
STANDBY_ARCHIVE_ DEST	NULL	Standby database archivelog destination text string
STAR_ TRANSFORMATION_ ENABLED	FALSE	Enable the use of star transformation
TAPE_ASYNCH_IO	TRUE	Use asynch I/O requests for tape devices

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	VALUE	DESCRIPTION
TEXT_ENABLE	TRUE	Enable text searching
THREAD	0	Redo thread to mount (OPS)
TIMED_OS_STATISTICS	0	Internal os statistic gathering interval in seconds
TIMED_STATISTICS	FALSE	Maintain internal timing statistics
TRANSACTION_AUDITING	TRUE	Auditing transaction generated in the redo log
TRANSACTIONS	1.1*sessions	Max. number of concurrent active transactions
TRANSACTIONS_PER_ROLLBACK_SEGMENT	5	Number of active transactions per rollback segment
USE_INDIRECT_DATA_BUFFERS	FALSE	Enable indirect data Buffers (very large SGA on 32-bit platforms)
USER_DUMP_DEST	(PD)	User process dump directory
UTL_FILE_DIR	NULL	Utl_file accessible directories list

Table 11: Oracle8i initialization parameters.

The Oracle7 and Oracle8.0 initialization parameters shown in Table 12 are invalid for use with Oracle8i.

NAME
ALLOW_PARTIAL_SN_RESULTS
ARCH_IO_SLAVES
B_TREE_BITMAP_PLANS
BACKUP_DISK_IO_SLAVES

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME
CACHE_SIZE_THRESHOLD
CLEANUP_ROLLBACK_ENTRIES
CLOSE_CACHED_OPEN_CURSORS
COMPATIBLE_NO_RECOVERY
DB_BLOCK_CHECKPOINT_BATCH
DB_BLOCK_LRU_EXTENDED_STATISTICS
DB_BLOCK_LRU_STATISTICS
DB_FILE_SIMULTANEOUS_WRITES
DISCRETE_TRANSACTIONS_ENABLED
DISTRIBUTED_LOCK_TIMEOUT
DISTRIBUTED_RECOVERY_CONNECTION_HOLD_TIME
DISTRIBUTED_TRANSACTIONS
FAST_FULL_SCAN_ENABLED
GC_DB_LOCKS
GC_FREELIST_GROUPS
GC_LATCHES
GC_LCK_PROCS
GC_ROLLBACK_SEGMENTS
GC_SAVE_ROLLBACK_LOCKS
GC_SEGMENTS
GC_TABLESPACES
LARGE_POOL_MIN_ALLOC
LGWR_IO_SLAVES
LOCK_SGA_AREAS
LOG_ARCHIVE_BUFFER_SIZE
LOG_ARCHIVE_BUFFERS
LOG_BLOCK_CHECKSUM
LOG_SIMULTANEOUS_COPIES
LOG_SMALL_ENTRY_MAX_SIZE
MAX_TRANSACTION_BRANCHES
MTS_LISTENER_ADDRESS
MTS_MULTIPLE_LISTENERS
MTS_RATE_LOG_SIZE
MTS_RATE_SCALE

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME
MTS_SERVICE
OGMS_HOME
OPS_ADMIN_GROUP
PARALLEL_DEFAULT_MAX_INSTANCES
PUSH_JOIN_PREDICATE
REDUCE_ALARM
SEQUENCE_CACHE_ENTRIES
SEQUENCE_CACHE_HASH_BUCKETS
SHARED_POOL_RESERVED_MIN_ALLOC
SORT_READ_FAC
SORT_SPACEMAP_SIZE
SORT_WRITE_BUFFER_SIZE
SORT_WRITE_BUFFERS
SPIN_COUNT
TEMPORARY_TABLE_LOCKS

Table 12: Invalid initialization parameters for use with Oracle8i.

The DBA should review the applicable Database Administrator's and Tuning Guides before modifying any INIT.ORA parameters.

Oracle 8i has expanded the number of initialization parameters that can be altered by use of the ALTER SESSION, ALTER SYSTEM SET and ALTER SYSTEM SET … DEFERRED commands. Table 13 shows the initialization parameters that can be altered with the above commands.

NAME	ALTER SESSION	ALTER SYSTEM	DEFERRED
AQ_TM_PROCESSES		X	
BACKGROUND_DUMP_ DEST		X	
BACKUP_TAPE_IO_ SLAVES		X	

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	ALTER SESSION	ALTER SYSTEM	DEFERRED
CONTROL_FILE_ RECORD_KEEP_TIME		X	
CORE_DUMP_DEST		X	
DB_BLOCK_CHECKSUM		X	
DB_BLOCK_CHECKING	X		X
DB_BLOCK_MAX_ DIRTY_TARGET		X	
DB_FILE_MULTIBOCK_ READ_COUNT	X	X	
DB_FILE_DIRECT_IO_ COUNT			X
FIXED_DATE		X	
FREEZE_DB_FOR_ FAST_INSTANCE_ RECOVERY		X	
GC_DEFER_TIME		X	
GLOBAL_NAMES	X	X	
HASH_AREA_SIZE	X		
HASH_JOIN_ENABLED	X		
HASH_MULTIBLOCK_IO_ COUNT	X	X	
HS_AUTOREGISTER		X	
JOB_QUEUE_ PROCESSES		X	
LICENSE_MAX_ SESSIONS		X	
LICENSE_MAX_USERS		X	
LICENSE_SESSIONS_ WARNING		X	
LOG_ARCHIVE_DEST		X	
LOG_ARCHIVE_DEST_n	X	X	
LOG_ARCHIVE_DEST_ STATE_n	X	X	

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	ALTER SESSION	ALTER SYSTEM	DEFERRED
LOG_ARCHIVE_ DUPLEX_DEST		X	
LOG_ARCHIVE_MAX_ PROCESSES		X	
LOG_ARCHIVE_MIN_ SUCCEED_DEST	X	X	
LOG_CHECKPOINT_ INTERVAL		X	
LOG_CHECKPOINT_ TIMEOUT		X	
MAX_DUMP_FILE_SIZE	X	X	
MTS_DISPATCHERS		X	
MTS_SERVERS		X	
MVIEW_REWRITE_ ENABLED	X	X	
NLS_CALENDAR	X		
NLS_COMP	X		
NLS_CURRENCY	X		
NLS_DATE_FORMAT	X		
NLS_DATE_LANGUAGE	X		
NLS_ISO_CURRENCY	X		
NLS_LANGUAGE	X		
NLS_NUMERIC_ CHARACTERS	X		
NLS_SORT	X		
NLS_TERRITORY	X		
NLS_UNION_ CURRENCY	X		
OBJECT_CACHE_MAX_ SIZE_PERCENT	X		X
OBJECT_CACHE_ OPTIMAL_SIZE	X		X
OPS_ADMIN_GROUP	X		

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	ALTER SESSION	ALTER SYSTEM	DEFERRED
OPTIMIZER_INDEX_ CACHING	X		
OPTIMIZER_INDEX_ COST_ADJ	X		
OPTIMIZER_MAX_ PERMUTATIONS	X		
OPTIMIZER_MODE	X		
OPTIMIZER_PERCENT_ PARALLEL	X		
OPTIMIZER_SEARCH_ LIMIT	X		
PARALLEL_ADAPTIVE_ MULTI_USER		X	
PARALLEL_ BROADCAST_ENABLED	X		
PARALLEL_INSTANCE_ GROUP	X	X	
PARALLEL_MIN_ PERCENT	X		
PARALLEL_THREADS_ PER_CPU		X	
PARALLEL_ TRANSACTION_ RECOVERY		X	
PARTITION_VIEW_ ENABLED	X		
PLSQL_V2_ COMPATIBILITY	X	X	
REMOTE_ DEPENDENCIES_MODE	X	X	
RESOURCE_LIMIT		X	
RESOURCE_MANAGER_ PLAN		X	

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	ALTER SESSION	ALTER SYSTEM	DEFERRED
REWRITE_INTEGRITY	X	X	
SESSION_CACHED_ CURSORS	X		
SORT_AREA_RETAINED _SIZE	X		X
SORT_AREA_SIZE	X		X
SORT_MULTIBLOCK_ READ_COUNT	X		X
STANDBY_ARCHIVE_ DEST		X	
STAR_ TRANSFORMATIONS_ ENABLED	X		
TEXT_ENABLE	X	X	
TIMED_OS_STATISTICS		X	
TIMED_STATISTICS	X	X	
TRANSACTION_ AUDITING			X
USER_DUMP_DEST		X	

Table 13: Initialization Parameters Which Can be Altered During Operation

While the ALTER SYSTEM command will change the value of the initialization parameter for the instance, it may not change it for all sessions. The DEFERRED clause only alters the parameter for new sessions, not existing sessions. There are some initialization parameters specific to the heterogeneous services option for Oracle8 and 8i, these are altered using the DBMS_HS package and are not all listed above.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The Undocumented Initialization Parameters ("_")

In addition to the Oracle-documented initialization parameters, there are varying numbers of undocumented initialization parameters in every version of Oracle. These undocumented initialization parameters are usually only used in emergencies and should only be used under the direction of a senior DBA or Oracle support. Source 6 shows a script for getting the undocumented initialization parameters out of an Oracle 7.2 instance and then out of a 7.3 or 8.0.2 instance.

SOURCE 6: Scripts for documenting undocumented initialization parameters

```
(pre-7.3 and post-7.3).
REM Script for getting undocumented init.ora
REM parameters from a 7.2 instance
REM MRA - Revealnet 2/23/97
REM
COLUMN parameter FORMAT a40
COLUMN value FORMAT a30
COLUMN ksppidf HEADING 'Is|Default'
SET FEEDBACK OFF VERIFY OFF PAGES 55
START title80 'Undocumented Init.ora Parameters'
SPOOL rep_out/&db/undoc
SELECT  ksppinm "Parameter",
  ksppivl "Value",
  ksppidf
FROM x$ksppi
WHERE ksppinm LIKE '/_%' escape '/'
/
SPOOL OFF
TTITLE OFF

REM Script for getting undocumented init.ora
REM parameters from a 7.3, 8.0.x, or 8.1.3 instance
REM MRA - Revealnet 4/23/97
REM
COLUMN parameter           FORMAT a37
COLUMN description      FORMAT a30 WORD_WRAPPED
COLUMN "Session Value"      FORMAT a10
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
COLUMN "Instance Value" FORMAT a10
SET LINES 100
SET PAGES 0
SPOOL undoc.lis
SELECT
     a.ksppinm  "Parameter",
     a.ksppdesc "Description",
     b.ksppstvl "Session Value",
     c.ksppstvl "Instance Value"
FROM
     x$ksppi a,
     x$ksppcv b,
     x$ksppsv c
WHERE
     a.indx = b.indx
     AND a.indx = c.indx
     AND a.ksppinm LIKE '/_%' escape '/'
/
SPOOL OFF
SET LINES 80 PAGES 20
CLEAR COLUMNS
```

The output for an Oracle 7.2 database looks like Listing 13.

LISTING 13: Undocumented initialization parameters for Oracle 7.2 not included in Oracle 7.3 and 8.0.2.

```
Date: 03/04/97
Page:   1
Time: 03:47 PM                    Undocumented Init.ora
Parameters      SYS
                                        ORCNETD1 database

Is
Parameter                                Value
Default
-------------------------------------------------------------
------------------
_latch_spin_count (7.3 ¤ _spin_count)    100
TRUE
_trace_instance_termination              FALSE
TRUE
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
_wakeup_timeout                          100
TRUE
_lgwr_async_write                        TRUE
TRUE
```

The undocumented parameters for 7.3 are listed in Listing 13. Note that the descriptions for the parameters are available starting with Oracle 7.3. The undocumented parameters new for Oracle8 are shown in Listing 14.

LISTING 14: Undocumented initialization parameters for Oracle 7.3 not included in 8.0.5.

```
Parameter Name                   Parameter Description
Default Value
------------------------------------------------------------
-----------------
_standby_lock_name_space         lock name space used for
                                 generating lock names for
                                 standby datab

_enable_dba_locking              enable persistent locking
FALSE
```

Table 14 shows all of the undocumented initialization parameters for 8i.

Parameter	Value	Description
_NUMA_instance_ mapping	Not Specified	Set of nodes that this instance should run on
_NUMA_pool_size	Not Specified	aggregate size in bytes of NUMA pool
_advanced_dss_features	FALSE	enable advanced DSS features
_affinity_on	FALSE	enable/disable affinity at run time

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_all_shared_dblinks	NULL	treat all dblinks as shared
_allocate_creation_order	FALSE	should files be examined in creation order during allocation
_allow_read_only_ corruption	FALSE	allow read-only open even if database is corrupt
_allow_resetlogs_ corruption	FALSE	allow resetlogs even if it will cause corruption
_always_star_ transformation	FALSE	always favor use of star transformation
_arch_io_slaves	0	ARCH I/O slaves
_average_dirties_half_life	2097152	Half life in bytes of first dirties running average
_b_tree_bitmap_plans	FALSE	enable the use of bitmap plans for tables w. only B-tree indexes
_backup_disk_io_slaves	0	BACKUP Disk I/O slaves
_backup_io_pool_size	262144	memory to reserve from the large pool
_bump_highwater_mark_ count	0	How many blocks should we allocate per free list on advancing HW
_cleanup_rollback_entries	20	no. of undo entries to apply per transaction cleanup
_close_cached_open_ cursors	FALSE	close cursors cached by PL/SQL at each commit

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_column_elimination_off	FALSE	turn off predicate-only column elimination
_compatible_no_recovery	0.0.0	Database will be compatible unless crash or media recovery is needed
_complex_view_merging	FALSE	enable complex view merging
_controlfile_enqueue_ timeout	900	control file enqueue timeout in seconds
_corrupt_blocks_on_ stuck_recovery	0	number of times to corrupt a block when media recovery stuck
_corrupted_rollback_ segments	NULL	corrupted undo segment list
_cpu_to_io	NULL	multiplier for converting CPU cost to I/O cost
_cr_server	NULL	if TRUE, enable CR server TRUE (DFS)
_cursor_db_buffers_ pinned	78	additional number of buffers a cursor can pin at once
_db_aging_cool_count	1	Touch count set when buffer cooled
_db_aging_freeze_cr	2	Make CR buffers always be FALSE too cold to keep in cache
_db_aging_hot_criteria	2	Touch count which sends a buffer to head of replacement list
_db_aging_stay_count	99	Touch count set when buffer moved to head of replacement list

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_db_aging_touch_time	3	Touch count which sends a buffer to head of replacement list
_db_block_cache_clone	FALSE	Always clone data blocks on get (for debugging)
_db_block_cache_map	0	Map / unmap and track reference counts on blocks (for debugging)
_db_block_cache_protect	FALSE	protect database blocks (true only when debugging)
_db_block_hash_buckets	2000	Number of database block hash buckets
_db_block_hi_priority_ batch_size	0	Fraction of writes for high priority reasons
_db_block_max_cr_dba	10	Maximum Allowed Number of CR buffers per dba
_db_block_max_scan_cnt	0	Maximum number of buffers to inspect when looking for free
_db_block_med_priority_ batch_size	0	Fraction of writes for medium priority reasons
_db_handles	700	System-wide simultaneous buffer operations
_db_handles_cached	5	Buffer handles cached each process
_db_large_dirty_queue	0	Number of buffers which force dirty queue to be written

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_db_no_mount_lock	FALSE	do not get a mount lock
_db_percent_hot_default	50	Percent of default buffer pool considered hot
_db_percent_hot_keep	0	Percent of keep buffer pool considered hot
_db_percent_hot_recycle	0	Percent of recycle buffer pool considered hot
_db_writer_chunk_writes	0	Number of writes DBWR should wait for
_db_writer_histogram_ statistics	FALSE	maintain dbwr histogram statistics in x$kcbbhs
_db_writer_max_writes	0	Max number of outstanding DB Writer Ios
_db_writer_scan_depth	0	Number of LRU buffers for dbwr to scan when looking for dirty blocks
_db_writer_scan_depth_ decrement	0	Subtract from dbwr scan depth when dbwr is working too hard
_db_writer_scan_depth_ increment	0	Add to dbwr scan depth when dbwr is behind
_dbwr_async_io	TRUE	Enable dbwriter asynchronous writes
_dbwr_tracing	0	Enable dbwriter tracing

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_defer_multiple_waiters	TRUE	if TRUE, defer down converts when there were waiters (DFS)
_disable_incremental_ checkpoints	FALSE	Disable incremental checkpoints for thread recovery
_disable_latch_free_ SCN_writes_via_32cas	FALSE	disable latch-free SCN writes using 32-bit compare & swap
_disable_latch_free_SCN_ writes_via_64cas	TRUE	disable latch-free SCN writes using 64-bit compare & swap
_disable_logging	FALSE	Disable logging
_disable_ntlog_events	FALSE	Disable logging to NT event log
_discrete_transactions_ enabled	FALSE	enable OLTP mode
_distributed_lock_timeout	60	number of seconds a distributed transaction waits for a lock
_distributed_recovery_ connection_hold_time	200	number of seconds RECO holds outbound connections open
_domain_index_batch_ size	2000	maximum number of rows from one call to domain index fetch routine
_domain_index_dml_ batch_size	200	maximum number of rows for one call to domain index dml routines
_dss_cache_flush	TRUE	enable full cache flush for parallel execution

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_dynamic_stats_threshold	30	Delay threshold (in seconds) between sending statistics messages
_enable_NUMA_ optimization	TRUE	Enable NUMA specific optimizations
_enable_block_level_ transaction_recovery	true	enable block level recovery
_enable_cscn_caching	TRUE	enable commit SCN caching for all transactions
_enable_default_affinity	0	to enable default implementation of affinity osds
_enable_list_io	TRUE	Enable List I/O
_enable_multitable_ sampling	FALSE	enable multitable sampling
_enqueue_debug_multi_ instance	FALSE	Debug enqueue multi-instance
_enqueue_hash	265	enqueue hash table length
_enqueue_hash_chain_ latches	2	enqueue hash chain latches
_enqueue_locks	1479	Locks for managed enqueues
_fairness_threshold	4	number of times to CR serve before downgrading lock (DFS)
_fast_full_scan_enabled	FALSE	enable fast full scan
_foreground_locks	TRUE	if TRUE, foregrounds acquire PCM locks (DFS)
_full_pwise_join_enabled	TRUE	enable full partition-wise join when TRUE

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_gc_class_locks	0	set locks for the minor classes (DFS)
_gc_latches	4	number of latches per lock process (DFS)
_gc_lck_procs	1	number of background parallel server lock processes to start
_groupby_nopushdown_ cut_ratio	3	groupby nopushdown cut ratio
_high_server_threshold	0	high server thresholds
_idl_conventional_index_ maintenance	TRUE	enable conventional index maintenance for insert direct load
_index_prefetch_factor	100	Index prefetching factor
_init_sql_file	%ORACLE%\ RDBMS\ADM IN\SQL.BSQ	File containing SQL statements to execute upon database creation
_io_slaves_disabled	FALSE	Do not use I/O slaves
_ioslave_batch_count	1	Per attempt IOs picked
_ioslave_issue_count	500	IOs issued before completion check
_kcl_debug	FALSE	if TRUE, record le history (DFS)
_kgl_bucket_count	0	Index to the bucket count array
_kgl_latch_count	0	number of library cache latches
_kgl_multi_instance_ invalidation	TRUE	whether KGL to support multi-instance invalidations

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_kgl_multi_instance_lock	TRUE	whether KGL to support multi-instance locks
_kgl_multi_instance_pin	TRUE	whether KGL to support multi-instance pins
_kkfi_trace	FALSE	Trace expression substitution
_large_pool_min_alloc	16K	minimum allocation size in bytes for the large allocation pool
_latch_miss_stat_sid	0	Sid of process for which to collect latch stats
_latch_recovery_ alignment	254	Align latch recovery structures
_latch_wait_posting	1	post sleeping processes when free latch
_lazy_freelist_close	TRUE	if TRUE, locks on the freelist are closed (DFS)
_left_nested_loops_ random	FALSE	enable random distribution method for left of nestedloops
_lgwr_async_io	TRUE	LGWR Asynchronous IO enabling boolean flag
_lgwr_io_slaves	0	LGWR I/O slaves
_lm_ast_option	FALSE	enable ast passthrough option
_lm_direct_sends	ALL	Processes which will do direct lock manager sends

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_lm_dlmd_procs	1	number of background lock manager daemon processes to start
_lm_domains	2	number of groups configured for the lock manager
_lm_groups	20	number of groups configured for the lock manager
_lm_non_fault_tolerant	FALSE	disable lock manager fault-tolerance mode
_lm_send_buffers	10000	number of lock manager send buffers
_lm_statistics	FALSE	enable lock manager statistics collection
_lm_xids	139	number of transaction IDs configured for the lock manager
_lock_sga_areas	0	Lock specified areas of the SGA in physical memory
_log_archive_buffer_size	127	Size of each archival buffer in log file blocks
_log_archive_buffers	4	Number of buffers to allocate for archiving
_log_blocks_during_ backup	TRUE	log block images when changed during backup
_log_buffers_debug	FALSE	Debug redo buffers (slows things down)
_log_checkpoint_ recovery_check	0	# redo blocks to verify after checkpoint
_log_debug_multi_ instance	FALSE	Debug redo multi instance code

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_log_io_size	0	automatically initiate log write if this many redo blocks in buffers
_log_simultaneous_copies	4	number of simultaneous copies into redo buffer(#of copy latches)
_log_space_errors	TRUE	should we report space errors to alert log
_low_server_threshold	0	low server thresholds
_max_exponential_sleep	0	max sleep during exponential backoff
_max_sleep_holding_latch	4	max time to sleep while holding a latch
_max_transaction_ branches	8	max. number of branches per distributed transaction
_messages	200	message queue resources - dependent on # processes & # buffers
_minimum_giga_scn	0	Minimum SCN to start with in 2^30 units
_mts_load_constants	300,19,64,3,1 0,10,0	server load balancing constants (S,P,D,I,L,C,M)
_mts_rate_log_size	NULL	number of rate statistic events to log
_mts_rate_scale	NULL	Scale to display rate statistic (100ths of a second)
_mview_cost_rewrite	TRUE	perform the cost based rewrite with mviews

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_mview_rewrite_2	TRUE	perform mview rewrite before&after or only after view merging
_ncmb_readahead_ enabled	0	enable multi-block readahead for an index scan
_ncmb_readahead_tracing	FALSE	turn on multi-block readahead tracing
_nested_loop_fudge	100	nested loop fudge
_no_objects	FALSE	no object features are used
_no_or_expansion	FALSE	OR expansion during optimization disabled
_number_cached_ attributes	10	maximum number of cached attributes per instance
_offline_rollback_ segments	NULL	offline undo segment list
_ogms_home	NULL	GMS home directory
_open_files_limit	4294967294	Limit on number of files opened by I/O subsystem
_optimizer_undo_changes	FALSE	undo changes to query optimizer
_oracle_trace_events	NULL	Oracle TRACE event flags
_oracle_trace_facility_ version	NULL	Oracle TRACE facility version
_ordered_nested_loop	FALSE	enable ordered nested loop costing
_parallel_adaptive_max_ users	1	maximum number of users running with a default DOP

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_parallel_default_max_ instances	1	default maximum number of instances for parallel query
_parallel_execution_ message_align	FALSE	message buffer size parallel execution
_parallel_fake_class_pct	0	fake db-scheduler percent used for testing
_parallel_load_bal_unit	0	number of threads to allocate per instance
_parallel_load_balancing	TRUE	parallel execution load balanced slave allocation
_parallel_min_message_ pool	122880	minimum size of shared pool memory to reserve for pq servers
_parallel_recovery_stopat	32767	stop at -position- to step through SMON
_parallel_server_idle_time	5	idle time before parallel query server dies
_parallel_server_sleep_ time	10	Sleep time between dequeue timeouts (in 1/100ths)
_parallelism_cost_fudge_ factor	350	set the parallelism cost fudge factor
_partial_pwise_join_ enabled	TRUE	enable partial partition-wise join when TRUE
_passwordfile_enqueue_ timeout	900	password file enqueue timeout in seconds

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_pdml_separate_gim	TRUE	prevent separation of global index maintenance for PDML
_pdml_slaves_diff_part	TRUE	slaves start on different partition when doing index maint
_plsql_dump_buffer_ events	NULL	conditions upon which the PL/SQL circular buffer is dumped
_predicate_elimination_ enabled	TRUE	Allow predicate elimination if set to TRUE
_project_view_columns	TRUE	enable projecting out unreferenced columns of a view
_push_join_predicate	FALSE	enable pushing join predicate inside a view
_px_broadcast_fudge_ factor	100	set the tq broadcasting fudge factor percentage
_px_broadcast_trace	FALSE	dump trace information about decisions as to when to broadcast
_px_granule_size	100000	default size of a rowid range granule (in KB)
_px_index_sampling	200	parallel query sampling for index create (100000= 100%)
_px_kxib_tracing	0	turn on kxib tracing

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_px_load_publish_interval	3000	interval at which LMON will check whether to publish PX load
_px_max_granules_per_ slave	100	maximum number of rowid range granules to generate per slave
_px_min_granules_per_ slave	13	minimum number of rowid range granules to generate per slave
_px_no_stealing	FALSE	prevent parallel granule stealing in shared nothing environment
_release_insert_threshold	5	maximum number of unusable blocks to unlink from freelist
_reuse_index_loop	5	number of blocks being examine for index block reuse
_rollback_segment_count	0	number of undo segments
_rollback_segment_initial	1	Starting undo segment number
_row_cache_buffer_size	200	size of row cache circular buffer
_row_cache_cursors	10	number of cached cursors for row cache management
_row_cache_instance_locks	100	number of row cache instance locks
_save_escalates	TRUE	if TRUE, save escalates from basts (DFS)
_scn_scheme	NULL	SCN scheme

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_serial_direct_read	FALSE	enable direct read in serial
_session_idle_bit_latches	0	one latch per session or a latch per group of sessions
_shared_pool_reserved_ min_alloc	4400	Minimum allocation size in bytes for reserved area of shared pool
_single_process	FALSE	run without detached processes
_small_table_threshold	160	threshold level of table size for forget-bit enabled during scan
_sort_space_for_write_ buffers	1	tenths of sort_area_size devoted to direct write buffers
_spin_count	2000	Amount to spin waiting for a latch
_sql_connect_capability_ override	0	SQL Connect Capability Table Override
_sql_connect_capability_ table	NULL	SQL Connect Capability Table (testing only)
_system_trig_enabled	TRUE	system triggers are enabled
_test_param_1	25	test parmeter 1
_test_param_2	0	test parameter 2
_test_param_3	0	test parameter 3
_tq_dump_period	0	time period for duping of TQ statistics(s)

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_trace_archive_dest	%ORACLE_HOME%\RDBMS\TRACE.DAT	trace archival destination
_trace_archive_start	FALSE	start trace process on SGA initialization
_trace_block_size	2048	trace block size
_trace_buffer_flushes	FALSE	trace buffer flushes if otrace cacheIO event is set
_trace_buffers_per_process	0	trace buffers per process
_trace_cr_buffer_creates	FALSE	trace cr buffer creates if otrace cacheIO event is set
_trace_enabled	TRUE	Should tracing be enabled at startup
_trace_events	NULL	turns on and off trace events
_trace_file_size	10000	trace file size
_trace_files_public	FALSE	Create publicly accessible trace files
_trace_flushing	FALSE	TRWR should try to keep tracing buffers clean
_trace_multi_block_reads	FALSE	Trace multi_block reads if otrace cacheIO event is set
_trace_write_batch_size	32	trace write batch size
_transaction_recovery_servers	0	max number of parallel recovery slaves that may be used
_use_ism	TRUE	Enable Shared Page Tables – ISM

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Parameter	Value	Description
_use_vector_post	TRUE	use vector post
_wait_for_sync	TRUE	wait for sync on commit MUST BE ALWAYS TRUE
_walk_insert_threshold	0	maximum number of unusable blocks to walk across freelist
_yield_check_interval	5	Interval to check whether actses should yield

* Indicates that this parameter is Oracle8 only.

TABLE 14: Undocumented initialization parameters for 8i.

You will note that each undocumented parameter begins with an underscore (_) character. For those of you who have been around awhile, you will also notice some of these “undocumented” parameters used to be documented. Some you may have seen used, such as “_offline_rollback_segments”; others you will never use or see used. However, you should be aware that there are more parameters than those listed in a user’s manual. You may need to prompt Oracle support if you see one (such as “_corrupted_rollback_segments”) that may just be helpful in a sticky situation. Another set of parameters that may be useful are events. Let’s discuss events a bit.

The Initialization File Event Settings

The SET EVENTS command in an init<SID>.ora file has generally been placed there at the command of Oracle support to perform specific functions. Usually these alerts turn on more advanced levels of tracing and error detection than are commonly available. Source 7 lists some of the more common events.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

SOURCE 7: Example uses of Oracle event codes.

```
To enable block header and trailer checking to detect
corrupt blocks:
event="10210 trace name context forever, level 10"  ¤ for
tables
event="10211 trace name context forever, level 10"  ¤ for
indexes

and to go with these the undocumented parameter setting:
 _db_block_cache_protect=TRUE

Which will prevent corruption from getting to your disks
(at the cost of a database crash)


For tracing of a MAX_CURSORS exceeded error:

event="1000 trace name ERRORSTACK level 3"

To get an error stack related to a SQLNET ORA-03120 error:

event="3120 trace name error stack"

To turn on bitmapped indexes  in 7.3.2:

event="10111 trace name context forever"
event="10112 trace name context forever"
event="10114 trace name context forever"

and˜set the initialization parameter COMPATIBLE to 7.3.2

To work around a space leak problem:

event="10262 trace name context forever, level 1024"

To trace memory shortages:

event="10235 trace name context forever, level 4"
event="600 trace name heapdump, level 4"

Events are also used as the SESSION level:
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
alter session set events '10046 trace name context forever
level NN'
     where NN:
          1 " same as a regular trace
          4 " also dump bind variables
          8 " also dump wait information
          12 " dump both bind and wait information

To coalesce freespace in a tablespace pre-7.3:

alter session set events 'immediate trace name coalesce
level XX'
     where XX:
                the value of ts# from ts$ table for the
tablespace
```

A list of the Oracle Event codes is shown in Table 15.

Code	Description
10000	Control file debug event, name 'control_file'
10001	Control file crash event1
10002	Control file crash event2
10003	Control file crash event3
10004	Control file crash event4
10005	Trace latch operations for debugging
10006	Testing - block recovery forced
10007	Log switch debug crash after new log select, thread %s
10008	Log switch debug crash after new log header write, thread %s
10009	Log switch debug crash after old log header write, thread %s
10010	Begin Transaction
10011	End Transaction
10012	Abort Transaction
10013	Instance Recovery
10014	Roll Back to Save Point
10015	Undo Segment Recovery
10016	Undo Segment extend

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Code	Description
10017	Undo Segment Wrap
10018	Data Segment Create
10019	Data Segment Recovery
10020	Partial link restored to linked list (KSG)
10021	Latch cleanup for state objects (KSS)
10022	Trace ktsgsp
10023	Create Save Undo Segment
10024	Write to Save Undo
10025	Extend Save Undo Segment
10026	Apply Save Undo
10027	Latch cleanup for enqueue locks (KSQ)
10028	Latch cleanup for enqueue resources (KSQ)
10029	Session logon (KSU)
10030	Session logoff (KSU)
10031	Row source debug event (R*)
10032	Sort end (SOR*)
10035	Parse SQL statement (OPIPRS)
10036	Create remote row source (QKANET)
10037	Allocate remote row source (QKARWS)
10038	Dump row source tree (QBADRV)
10039	Type checking (OPITCA)
10040	Dirty cache list
10041	Dump undo records skipped
10042	Trap error during undo application
10044	Free list undo operations
10045	Free list update operations - ktsrsp, ktsunl
10046	Enable SQL statement timing
10047	Trace switching of sessions
10048	Undo segment shrink
10049	Protect library cache memory heaps
10050	Sniper trace
10051	Trace OPI calls
10052	Don’t clean up obj$
10053	CBO Enable optimizer trace

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Code	Description
10054	Trace UNDO handling in MLS
10055	Trace UNDO handing
10056	Dump analyze stats (kdg)
10057	Suppress file names in error messages
10058	Use table scan cost in tab$.spare1
10060	CBO Enable predicate dump
10061	Disable SMON from cleaning temp segment
10062	Disable usage of OS Roles in osds
10063	Disable usage of DBA and OPER privileges in osds
10064	Thread enable debug crash level %s, thread %s
10065	Limit library cache dump information for state object dump
10066	Simulate failure to verify file
10067	Force redo log checksum errors - block number
10068	Force redo log checksum errors - file number
10069	Trusted Oracle test event
10070	Force datafile checksum errors - block number
10071	Force datafile checksum errors - file number
10072	Protect latch recovery memory
10073	Have PMON dump info before latch cleanup
10074	Default trace function mask for kst
10075	CBO Disable outer-join to regular join conversion
10076	CBO Enable cartesian product join costing
10077	CBO Disable view-merging optimization for outer-joins
10078	CBO Disable constant predicate elimination optimization
10080	Dump a block on a segment list which cannot be exchanged
10081	Segment High Water Mark has been advanced
10082	Free list head block is the same as the last block
10083	A brand new block has been requested from space management
10084	Free list becomes empty
10085	Free lists have been merged
10086	CBO Enable error if kko and qka disagree on oby sort

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Code	Description
10087	Disable repair of media corrupt data blocks
10088	CBO Disable new NOT IN optimization
10089	CBO Disable index sorting
10090	Invoke other events before crash recovery
10091	CBO Disable constant predicate merging
10092	CBO Disable hash join
10093	CBO Enable force hash joins
10094	Before resizing a data file
10095	Dump debugger commands to trace file
10096	After the cross instance call when resizing a data file
10097	After generating redo when resizing a data file
10098	After the OS has increased the size of a data file
10099	After updating the file header with the new file size
10100	After the OS has decreased the size of a data file
10101	Atomic redo write recovery
10102	Switch off anti-joins
10103	CBO Disable hash join swapping
10104	Dump hash join statistics to trace file
10105	CBO Enable constant pred trans and MPs w WHERE-clause
10106	CBO Disable evaluating correlation pred last for NOT IN
10107	CBO Always use bitmap index
10108	CBO Don't use bitmap index
10109	CBO Disable move of negated predicates
10110	CBO Try index rowid range scans
10111	Bitmap index creation switch
10112	Bitmap index creation switch
10113	Bitmap index creation switch
10114	Bitmap index creation switch
10115	CBO Bitmap optimization use maximal expression
10116	CBO Bitmap optimization switch
10117	CBO Disable new parallel cost model
10118	CBO Enable hash join costing
10119	QKA Disable GBY sort elimination

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Code	Description
10120	CBO Disable index fast full scan
10121	CBO Don't sort bitmap chains
10122	CBO disable count(col) = count(*) transformation
10123	QKA Disable Bitmap And-EQuals
10145	Test auditing network errors
10146	Enable Oracle TRACE collection
10200	Block cleanout
10201	Consistent read undo application
10202	Consistent read block header
10203	Consistent read buffer status
10204	Signal recursive extend
10205	Row cache debugging
10206	Transaction table consistent read
10207	Consistent read transactions' status report
10208	Consistent read loop check
10209	Enable simulated error on control file
10210	Check data block integrity
10211	Check index block integrity
10212	Check cluster integrity
10213	Crash after control file write
10214	Simulate write errors on control file
10215	Simulate read errors on control file
10216	Dump control file header
10217	Debug sequence numbers
10218	Dump uba of applied undo
10219	Monitor multi-pass row locking
10220	Show updates to the transaction table
10221	Show changes done with undo
10222	Row cache
10223	Transaction layer - turn on verification codes
10226	Trace CR applications of undo for data operations
10227	Verify (multi-piece) row structure
10228	Trace application of redo by kcocbk
10230	Check redo generation by copying before applying

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Code	Description
10231	Skip corrupted blocks on _table_scans_
10232	Dump corrupted blocks symbolically when kcbgotten
10233	Skip corrupted blocks on index operations
10234	Trigger event after calling kcrapc to do redo N times
10235	Check memory manager internal structures
10236	Library cache manager
10237	Simulate ^C (for testing purposes)
10238	Instantiation manager
10239	Multi-instance library cache manager
10240	Dump dba's of blocks that we wait for
10241	Dump SQL generated for remote execution (OPIX)
10243	Simulated error for test %s of K2GTAB latch cleanup
10244	Make tranids in error msgs print as 0.0.0 (for testing)
10245	Simulate lock conflict error for testing PMON
10246	Print trace of PMON actions to trace file
10247	Turn on scgcmn tracing. (VMS ONLY)
10248	Turn on tracing for dispatchers
10249	Turn on tracing for multi-stated servers
10250	Trace all allocate and free calls to the topmost SGA heap
10251	Check consistency of transaction table and undo block
10252	Simulate write error to data file header
10253	Simulate write error to redo log
10254	Trace cross-instance calls
10256	Turn off multi-threaded server load balancing
10257	Trace multi-threaded server load balancing
10258	Force shared servers to be chosen round-robin
10259	Get error message text from remote using explicit call
10260	Trace calls to SMPRSET (VMS ONLY)
10261	Limit the size of the PGA heap
10262	Don't check for memory leaks
10263	Don't free empty PGA heap extents
10264	Collect statistics on context area usage (x$ksmcx)
10265	Keep random system generated output out of error messages

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Code	Description
10266	Trace OSD stack usage
10267	Inhibit KSEDMP for testing
10268	Don't do forward coalesce when deleting extents
10269	Don't do coalesces of free space in SMON
10270	Debug shared cursors
10271	Distributed transaction after COLLECT
10272	Distributed transaction before PREPARE
10273	Distributed transaction after PREPARE
10274	Distributed transaction before COMMIT
10275	Distributed transaction after COMMIT
10276	Distributed transaction before FORGET
10277	Cursor sharing (or not) related event (used for testing)
10281	Maximum time to wait for process creation
10282	Inhibit signalling of other backgrounds when one dies
10286	Simulate control file open error
10287	Simulate archiver error
10288	Do not check block type in ktrget
10289	Do block dumps to trace file in hex rather than fromatted
10290	Kdnchk - checkvalid event - not for general purpose use.
10291	Die in dtsdrv to test controlfile undo"
10292	Dump uet entries on a 1561 from dtsdrv"
10293	Dump debugging information when doing block recovery"
10294	Enable PERSISTENT DLM operations on non-compliant systems"
10300	Disable undo compatibility check at database open
10301	Enable LCK timeout table consistency check"
10320	Enable data layer (kdtgrs) tracing of space management calls"
10352	Report direct path statistics
10353	Number of slots
10354	Turn on direct read path for parallel query
10355	Turn on direct read path for scans
10356	Turn on hint usage for direct read
10357	Turn on debug information for direct path

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Code	Description
10374	Parallel query server interrupt (validate lock value)
10375	Turn on checks for statistics rollups
10376	Turn on table queue statistics
10377	Turn off load balancing
10379	Direct read for rowid range scans (unimplemented)
10380	Kxfp latch cleanup testing event
10381	Kxfp latch cleanup testing event
10382	Parallel query server interrupt (reset)
10383	Auto parallelization testing event
10384	Parallel dataflow scheduler tracing
10385	Parallel table scan range sampling method
10386	Parallel SQL hash and range statistics
10387	Parallel query server interrupt (normal)
10388	Parallel query server interrupt (failure)
10389	Parallel query server interrupt (cleanup)
10390	Trace parallel query slave execution
10391	Trace rowid range partitioning
10392	Parallel query debugging bits
10393	Print parallel query statistics
10394	Allow parallelization of small tables
10395	Adjust sample size for range table queues
10396	Circumvent range table queues for queries
10397	Suppress verbose parallel coordinator error reporting
10398	Enable timeouts in parallel query threads
10399	Use different internal maximum buffer size
10400	Turn on system state dumps for shutdown debugging
10500	Turn on traces for SMON
10510	Turn off SMON check to offline pending offline rollbacksegment
10511	Turn off SMON check to cleanup undo dictionary
10512	Turn off SMON check to shrink rollback segments
10600	Check cursor frame allocation
10602	Cause an access violation (for testing purposes)

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Code	Description
10603	Cause an error to occur during truncate (for testing purposes)
10604	Trace parallel create index
10605	Enable parallel create index by default
10606	Trace parallel create index
10607	Trace index rowid partition scan
10608	Trace create bitmap index
10610	Trace create index pseudo optimizer
10666	Do not get database enqueue name
10667	Cause sppst to check for valid process ids
10690	Set shadow process core file dump type (Unix only)
10691	Set background process core file type (Unix only)
10700	Alter access violation exception handler
10701	Dump direct loader index keys
10702	Enable histogram data generation
10703	Simulate process death during enqueue get
10704	Print out information about what enqueues are being obtained
10706	Print out information about instance lock manipulation
10707	Simulate process death for instance registration
10708	Print out Tracing information for skxf multi instance comms
10709	Enable parallel instances in create index by default
10710	Trace bitmap index access
10711	Trace bitmap index merge
10712	Trace bitmap index or
10713	Trace bitmap index and
10714	Trace bitmap index minus
10715	Trace bitmap index conversion to rowids
10800	Disable Smart Disk scan
10801	Enable Smart Disk trace
10802	Reserved for Smart Disk
10803	Write timing statistics on OPS recovery scan
10804	Reserved for ksxb

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Code	Description
10805	Reserved for row sort
10900	Extent manager fault insertion event #%s
10924	Import storage parse error ignore event
10925	Trace name context forever
10926	Trace name context forever
10927	trace name context forever
10928	trace name context forever
10999	do not get database enqueue name

Table 15: List of Oracle Event Codes

Source 7 shows how to use events

Source 7 Use of SET EVENTS at the session level

```
How to dump a segment header - by Don Burleson


set heading off;
spool dump_em.sql;

select
'alter session set events ''immediate trace name blockdump
level '||
to_char((header_file*16777216)+header_block)||''';'
from
dba_segments
where
segment_name = 'VBAP';

spool off;

cat dump_em.sql
@dump_em
```

The segment header block dump will be in the user session trace file.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

As with the undocumented initialization parameters, events should only be used under the direction of a senior-level DBA or Oracle support.

System Global Area

The global memory area, called the System or Shared Global Area (SGA) is an area of CPU memory that is reserved for Oracle use only. It contains buffers that are used to speed transaction throughput and help maintain the system integrity and consistency. No data is altered directly on the disk; it all passes through the SGA. The size and configuration of the SGA is defined by a file called the initialization file or INIT.ORA file, which can contain information on each type of buffer or shared pool area in the SGA.

SGA

Figure 6 shows the SGA for Oracle8 and Oracle8i. Notice the numerous areas contained in the SGA and remember that each of these areas may be further subdivided into other related areas. The V$SGASTAT view has 39 areas listed in both an Oracle8 and Oracle8i database without multi-threaded server or Oracle Parallel server present.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

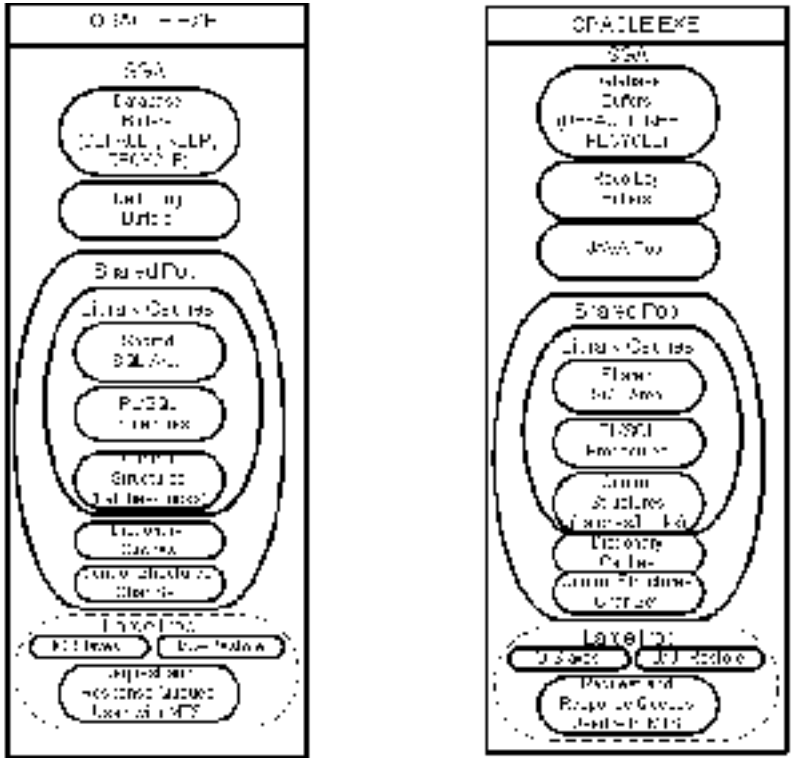


Figure 6: Oracle8 and Oracle8i SGAs

Virtually all of the areas in the SGA are configured either directly (SHARED_POOL_SIZE, DB_BLOCK_BUFFERS, LOG_BUFFERS, LARGE_POOL_SIZE, etc.) or indirectly using initialization parameters. Improperly sized SGA areas can cause large performance problems.

Modifying the INIT.ORA

The INIT.ORA (or initSID.ora) file sets the initialization parameters for an instance specified by the SID contained in the file name. To modify the parameters permanently you must alter them in the initialization file. As was shown previously there are numerous parameters which can be altered on the fly or on a per session basis using various forms of the ALTER command.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The values for your initialization parameters can be easily obtained by looking at the initialization file or by looking at the V$PARAMETER view. The SVRMGRL SHOW PARAMETERS command can also be used to show parameter values.

Any editor can be used to modify the initialization parameter file but it must be stored as a text file. Anytime a parameter is altered the effects of the change should be closely monitored. Invalid parameters will cause Oracle to not startup.

Allocating And Caching Memory

The name of the game in Oracle configuration is properly allocating and using memory. By using the initialization parameters we allocate memory for use by Oracle. We can determine how this memory is used by creating or altering objects using the CACHE/NOCACHE and the KEEP and RECYCLE POOL options. The DB Block Buffer area can be split into the DEFAULT, KEEP and RECYCLE pools that should be used to optimize how objects are cached.

In Oracle8 and Oracle8i the database block buffer has been split into three possible areas, the default, keep and recycle buffer pool areas. It is not required that these three pools be used, only one, the default pool configured with the DB_BLOCK_BUFFERS initialization parameter must be present, the others are “sub” pools to this main pool. How are the various pools used?

Use of the Default Pool

If a table, index or cluster is created without specifying that the KEEP or RECYCLE pool be used for its data, then it is placed in the default pool when it is accessed. This is standard Oracle7 behavior and if no special action is taken to use the other pools then this is also standard Oracle8 and Oracle8I behavior. The initialization parameters DB_BLOCK_BUFFERS and

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

DB_BLOCK_LRU_LATCHES must be set if multiple pools are to be used:

```
DB_BLOCK_BUFFERS = 2100
DB_BLOCK_LRU_LATCHES = 18
```

Use of The KEEP Pool

The KEEP database buffer pool is configured using the BUFFER_POOL_KEEP initialization parameter which looks like so:

```
BUFFER_POOL_KEEP = ‘300,3’
```

The two specified parameters are the number of buffers from the default pool to assign to the keep pool and the number of LRU (least recently used) latches to assign to the keep pool. The minimum number of buffers assigned to the pool is 50 times the number of assigned latches. The keep pool, as its name implies, is used to store object data that shouldn’t be aged out of the buffer pool such as look up information and specific performance enhancing indexes. The objects are assigned to the keep pool through either their creation statement or by specifically assigning them to the pool using the ALTER command. Any blocks already in the default pool are not affected by the ALTER command, only subsequently accessed blocks.

The keep pool should be sized such that it can hold all the blocks from all of the tables created with the buffer pool set to KEEP.

Use of the RECYCLE Pool

The RECYCLE database buffer pool is configured using the BUFFER_POOL_RECYCLE initialization parameter which looks like so:

```
BUFFER_POOL_RECYCLE = ‘900,6’
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The two specified parameters are the number of buffers from the default pool to assign to the recycle pool and the number of LRU (least recently used) latches to assign to the keep pool. The minimum number of buffers assigned to the pool is 50 times the number of assigned latches. The recycle pool, as its name implies, is used to store object data that should be aged out of the buffer pool rapidly such as searchable LOB information. The objects are assigned to the recycle pool through either their creation statement or by specifically assigning them to the pool using the ALTER command. Any blocks already in the default pool are not affected by the ALTER command, only subsequently accessed blocks.

As long as the recycle pool shows low block contention it is sized correctly.

With the above setpoints for the default, keep and recycle pools the default pool would end up with 900 buffers and 6 lru latches.

Tuning the Three Pools

Since the classic method of tuning the shared pool is not available in Oracle8i we must examine new methods to achieve the same ends. This involves looking at what Oracle has provided for tuning the new pools. A new script, **catperf.sql** offers several new views for tuning the Oracle buffer pools. These views are shown in Table 16.

V$BUFFER_POOL	Provides static information on pool configuration
V$BUFFER_POOL_STATISTICS	Provides Pool related statistics
V$DBWR_WRITE_HISTOGRAM	Provides summary information on DBWR write activities

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

V$DBWR_WRITE_LOG	Provides write information for each buffer area.

Table 16: Buffer Pool Related Views

Of the four new views the V$BUFFER_POOL_STATISTICS view seems the most useful for tuning the buffer pool. The V$BUFFER_POOL_STATISTICS view contains statistics such as **buffer_busy_waits, free_buffer_inspected, dirty_buffers_inspected** and physical write related data.

If a buffer pool shows excessive numbers of dirty_buffers_inspected and high amounts of buffer_busy_waits then it probably needs to be increased in size.

When configuring LRU latches and DBWR processes remember that the latches are assigned to the pools sequentially and to the DBRW processes round robin. The number of LRU processes should be equal to or a multiple of the value of DBWR processes to ensure that the DBRW load is balanced across the processes.

I encountered problems building the views as defined in the catperf.sql procedure for 8.1.3, hopefully by 8.1.5 (production) these problems are corrected.

Shared Pool

Let me begin this section by stating that the default values for the shared pool size initialization parameters are almost always too small by at least a factor of four. Unless your database is limited to the basic scott/tiger type schema and your overall physical data size is less than a couple of hundred megabytes, even the "large" parameters are far too small. What parameters control the size of the shared pool? Essentially only one, SHARED_POOL_SIZE. The other shared pool parameters control how the variable space areas in the shared pool are parsed out, but not overall shared pool size.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

In Oracle8 a new area, the large pool, controlled by the LARGE_POOL_SIZE parameter is also present. Generally speaking I suggest you start at a shared pool size of 40 megabytes and move up from there. The large pool size will depend on the number of concurrent users, number of multi-threaded server servers and dispatchers and the sort requirements for the application.

What should be monitored to determine if the shared pool is too small? For this you need to wade into the data dictionary tables, specifically the V$SGASTAT and V$SQLAREA views. Source 9. shows a report that shows how much of the shared pool is in use at any given time the script is run.

```
column shared_pool_used format 9,999.99
column shared_pool_size format 9,999.99
column shared_pool_avail format 9,999.99
column avail_pool_pct format 999.99
@title80 'Shared Pool Summary'
spool rep_out\&db\shared_pool
 select
  least(max(b.value)/(1024*1024),sum(a.bytes)/(1024*1024))
shared_pool_used,
  max(b.value)/(1024*1024) shared_pool_size,

greatest(max(b.value)/(1024*1024),sum(a.bytes)/(1024*1024))
-(sum(a.bytes)/(1024*1024)) shared_pool_avail,

((least(max(b.value)/(1024*1024),sum(a.bytes)/(1024*1024)))
/max(b.value))*100 avail_pool_pct
   from v$sgastat a, v$parameter b
 where (a.pool='shared pool'
 and a.name not in ('free memory'))
 and
 b.name='shared_pool_size';
rem SELECT
rem         SUM(a.BYTES)/1048576 pool_used,
rem                max(b.value)/(1024*1024)
shared_pool_size,
rem                (max(b.value)/(1024*1024))-
(sum(a.bytes)/(1024*1024)) shared_pool_avail,
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
rem                (sum(a.bytes)/max(b.value))*100
avail_pool_pct
rem   FROM
rem         v$sgastat a, v$parameter b
rem   WHERE
rem         a.name in (
rem         'reserved stopper',
rem         'table definiti',
rem         'dictionary cache',
rem         'library cache',
rem         'sql area',
rem         'PL/SQL DIANA',
rem         'SEQ S.O.') and
rem               b.name='shared_pool_size';
spool off
ttitle off
```

NOTE: In 8i a new column has been added to v$sgastat called POOL which can now be used to determine the size of the shared pool. However, in 8.1.3 a sum of the bytes for all objects in the shared pool exceeded the setpoint for shared_pool_size so this may not be as accurate as the above query. In fact, it indicated 7 megabytes free, 13 megabytes total with a setting of 10 megabytes.

Source 9 Example Script to Show SGA Usage

The script in Source 9 should be run periodically during times of normal and high usage of your database. The results will be similar to Listing 15. If your avail_pool_pct figures stay in the tens or below then you may need to increase the size of your shared pool, however, this isn't always the case.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
Date: 01/28/00
Page:   1
Time: 10:45 AM              Shared Pool Summary
MAULT
                                 dw database

SHARED_POOL_USED SHARED_POOL_SIZE SHARED_POOL_AVAIL
AVAIL_POOL_PCT
---------------- --------------- ----------------- --------
------
          597.46           619.89              22.43
.00

1 row selected.
```

Listing 15: Example Output From Script in Source 9.

To often all that is monitored is how much of the shared pool is filled, no one looks how is it filled; with good reusable SQL or bad, throw-away, garbage SQL. You must examine how the space is being used before you can decide whether the shared pool should be increased in size, decreased in size or perhaps a periodic flush schedule set up with the size remaining the same. So how can we determine what is in the shared pool and whether it is being properly reused or not? Let's look at a few more reports.

The first report we will examine shows how individual users are utilizing the shared pool. Before we can run the report a summary view of the V$SQLAREA view must be created, I unimaginatively call this view the SQL_SUMMARY view. The code for the SQL_SUMMARY view is shown in Source 10.

```
REM FUNCTION: Creates summary of v_$sqlarea and dba_users
for use in
REM           sqlmem.sql and sqlsummary.sql reports
REM   NOTE: Requires direct grants on v$sqlarea and
dba_users
REM
CREATE OR REPLACE VIEW sql_summary as
SELECT
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
   username, sharable_mem, persistent_mem, runtime_mem
FROM
   sys.v_$sqlarea a, dba_users b
WHERE
   a.parsing_user_id = b.user_id;
```

Source 10: Example SQL Script to Create A View to Monitor Pool Usage By User

Once the SQL_SUMMARY view is created the script in Source 11 is run to generate a summary report of SQL areas used by user. This shows the distribution of SQL areas and may show you that some users are hogging a disproportionate amount of the shared pool area. Usually, a user that is hogging a large volume of the shared pool is not using good SQL coding techniques that is generating a large number of non-reusable SQL areas.

```
REM
REM FUNCTION: Generate a summary of SQL Area Memory Usage
REM FUNCTION: uses the sqlsummary view.
REM           showing user SQL memory usage
REM
REM sqlsum.sql
REM
COLUMN areas                                HEADING
Used|Areas
COLUMN sharable   FORMAT 999,999,999      HEADING
Shared|Bytes
COLUMN persistent FORMAT 999,999,999      HEADING
Persistent|Bytes
COLUMN runtime    FORMAT 999,999,999      HEADING
Runtime|Bytes
COLUMN username   FORMAT a15              HEADING "User"
COLUMN mem_sum    FORMAT 999,999,999      HEADING Mem|Sum
START title80 "Users SQL Area Memory Use"
SPOOL rep_out\&db\sqlsum
SET PAGES 59 LINES 80
BREAK ON REPORT
COMPUTE SUM OF sharable ON REPORT
COMPUTE SUM OF persistent ON REPORT
COMPUTE SUM OF runtime ON REPORT
COMPUTE SUM OF mem_sum ON REPORT
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
SELECT
  username,
  SUM(sharable_mem) Sharable,
  SUM( persistent_mem) Persistent,
  SUM( runtime_mem) Runtime ,
  COUNT(*) Areas,
  SUM(sharable_mem+persistent_mem+runtime_mem) Mem_sum
FROM
  sql_summary
GROUP BY username
ORDER BY 2;
SPOOL OFF
PAUSE Press enter to continue
CLEAR COLUMNS
CLEAR BREAKS
SET PAGES 22 LINES 80
TTITLE OFF
```

Source 11: Example SQL Script To Report On SQL Area Usage By User

Example output from the script in Source 11 is shown in Listing 16. In the example report no one user is really hogging the SQL area. If you have a particular user that is hogging SQL areas, the script in Source 12 will show you what SQL areas they have and what is in them. This report on the actual SQL area contents can then be used to help teach the user how to better construct reusable SQL statements.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
Date: 01/28/00                                                    Page:   1
Time: 10:06 AM            Users SQL Area Memory Use               MAULT
                                dw database

                     Shared   Persistent      Runtime      Used
Mem
User                  Bytes        Bytes        Bytes     Areas          Sum
------------ ------------ ------------ ------------ --------- ------------
JSMITH        408,211,332   12,876,752   58,737,832     13814  479,825,916
SYS             7,458,619       86,912      350,088      2791    7,895,619
PRECISE        45,392,274    1,155,440   12,562,016       322   59,109,730
DWPROC          6,710,324      239,128    1,194,792       205    8,144,244
DSSUSER         4,985,220      174,304      742,136        97    5,901,660
NETSPO          5,907,293       86,032      657,384        51    6,650,709
GCMATCH         2,016,353       43,872      360,680        32    2,420,905
DCHUN           3,558,356       52,112      651,112        27    4,261,580
DWSYBASE          519,083       20,904      135,672        21      675,659
SMANN           1,040,273       18,368      114,648        12    1,173,289
MRCHDXD         1,228,587       18,656      131,352         9    1,378,595
MAULT             360,497        7,432       70,640         7      438,569
DATAWHSE          114,783        5,336       26,720         6      146,839
PATROL            218,605        5,232       35,480         6      259,317
SYSTEM            256,822        6,000       41,280         6      304,102
BKUEHNE            33,458        2,400        4,536         2       40,394
WPEREZ            243,786        4,560       31,824         2      280,170
TTIERNEY           41,518        2,160        4,320         2       47,998
CWOOD              61,856        2,952        6,960         2       71,768
MRCHAEM            24,744        2,312        6,912         2       33,968
JHUGHES            97,382        1,936        9,856         2      109,174
BUYRMRA            59,182        1,144        6,200         1       66,526
MRCHPWR            72,784        1,144        8,248         1       82,176
SSCOTT             80,585        1,032        9,664         1       91,281
TMANCEOR           21,570        1,536        2,384         1       25,490
SHOLETZ            76,901        1,112       13,160         1       91,173
MRCHPHP            57,445        1,624        7,680         1       66,749
INVCMTB           116,306        3,648       32,368         1      152,322
DEMERY             59,205        1,752        6,760         1       67,717
DBSNMP             14,416          816        5,840         1       21,072
            ------------ ------------ ------------ --------- ------------
sum           489,039,559   14,826,608   75,968,544     17427  579,834,711

30 rows selected.
```

Listing 16. Example Output From Source 11

In the example output we see that JSMITH user holds the most SQL areas and our SNMP user, DBSNMP holds the least. Usually the application owner will hold the largest section of memory in a well designed system, followed by ad-hoc users using properly designed SQL. In a situation where users aren't using properly

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

designed SQL statements the ad-hoc users will usually have the largest number of SQL areas and show the most memory usage as is shown in this example. Again, the script in Source 12 shows the actual in memory SQL areas for a specific user. Listng 17 shows the example output from a report run against GRAPHICS_USER in my test database using the script in Source 12.

```
rem
rem FUNCTION: Generate a report of SQL Area Memory Usage
rem           showing SQL Text and memory catagories
rem
rem sqlmem.sql
rem
COLUMN sql_text      FORMAT a60   HEADING Text word_wrapped
COLUMN sharable_mem               HEADING Shared|Bytes
COLUMN persistent_mem             HEADING Persistent|Bytes
COLUMN loads                      HEADING Loads
COLUMN users         FORMAT a15   HEADING "User"
COLUMN executions                 HEADING "Executions"
COLUMN users_executing            HEADING "Used By"
START title132 "Users SQL Area Memory Use"
SPOOL rep_out\&db\sqlmem
SET LONG 2000 PAGES 59 LINES 132
BREAK ON users
COMPUTE SUM OF sharable_mem ON USERS
COMPUTE SUM OF persistent_mem ON USERS
COMPUTE SUM OF runtime_mem ON USERS
SELECT
username users, sql_text, Executions, loads,
users_executing,
sharable_mem, persistent_mem
FROM
sys.v_$sqlarea a, dba_users b
WHERE
a.parsing_user_id = b.user_id
AND b.username LIKE UPPER('%&user_name%')
ORDER BY 3 DESC,1;
SPOOL OFF
PAUSE Press enter to continue
CLEAR COLUMNS
CLEAR COMPUTES
CLEAR BREAKS
SET PAGES 22 LINES 80
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Source 12: Example Script To Show Active SQL Areas For a User

```
Date: 11/18/98
Page:   1
Time: 04:19 PM                                        Users SQL
Area Memory Use                                      SYSTEM

ORTEST1 database
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
Shared  Pers
User    Text                                                  Executions Loads Used By  Bytes Bytes
------- ----------------------------------------------------- ---------- ----- ------- ------ -----
GRAPHICS_DBA BEGIN dbms_lob.read (:1, :2, :3, :4); END;            2121     1       0  10251   488
        alter session set nls_language= 'AMERICAN' nls_territory=     7     1       0   3975   408
         'AMERICA' nls_currency= '$' nls_iso_currency= 'AMERICA'
          nls_numeric_characters= '.,' nls_calENDar= 'GREGORIAN'
          nls_date_format= 'DD-MON-YY' nls_date_language= 'AMERICAN'
          nls_sort= 'BINARY'
          BEGIN :1 := dbms_lob.getLength (:2); END;                6     1       0   9290   448
          SELECT TO_CHAR(image_seq.nextval) FROM dual              6     1       0   6532   484
          SELECT graphic_blob FROM internal_graphics WHERE         2     1       0   5863   468
          graphic_id=10
          SELECT RPAD(TO_CHAR(graphic_id),5)||':                   1     1       0   7101   472
          '||RPAD(graphic_desc,30)||' : '||RPAD(graphic_type,10) FROM
           internal_graphics ORDER BY graphic_id
          SELECT graphic_blob FROM internal_graphics WHERE         1     1       0   6099   468
          graphic_id=12
          SELECT graphic_blob FROM internal_graphics WHERE         1     1       0   6079   468
          graphic_id=32
          SELECT graphic_blob FROM internal_graphics WHERE         1     1       0   6074   468
          graphic_id=4
          SELECT graphic_blob FROM internal_graphics WHERE         1     1       0   5962   468
          graphic_id=8
********** ------ -----
sum                                                                    67226  4640
```

Listing 17: Report Output Example For a Users SQL Area

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

One warning about the script in Source 12, the report it generates can run to several hundred pages for a user with a large number of SQL areas. What things should you watch for in a user's SQL areas?

- Watch for the non-use of bind variables, bind variable usage is shown by the inclusion of variables such as ":1" or ":B" in the SQL text.
- Watch for repetitive nearly identical SQL statements, notice the last several lines in the report in listing 17 for example.
- Watch for repetitive use of identical fixed SELECT statements, perhaps they can be moved into views.
- Watch for statements that are reloaded frequently

Notice that in the example report in Listing 17 the first four statements use bind variables, and, consequently are reusable. Non-bind usage means hard coded values such as 'Missing' or '10' are used. Notice that for most of the rest of the statements in the report no bind variables are used even though many of the SQL statements are nearly identical. This is one of the leading causes of shared pool misuse and results in useful SQL being drown in tons of non-reusable garbage SQL. Frequently reload of statements is an indication that the shared pool may be too small. Consider if the object showing reloads should be pinned into the SGA.

The problem with non-reusable SQL is that it must still be looked at by any new SQL inserted into the pool (actually its hash value is scanned). While a hash value scan may seem a small cost item, if your shared pool contains tens of thousands of SQL areas this can be a performance bottleneck. How can we determine, without running the report in Source 12 for each of possibly hundreds of users, if we have garbage SQL in the shared pool?

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The script in Source 13 shows a view that provides details on individual user's SQL area reuse. The view can be tailored to your environment if the limit on reuse (currently set at 1) is too restrictive. For example, in a recent tuning assignment resetting the value to 12 resulting in nearly 70 percent of the SQL being rejected as garbage SQL, in DSS or data warehouse systems where rollups are performed by the month, bi-monthly or weekly, values of 12, 24 or 52 might be advisable. Source 14 shows a report script that uses the view created in Source 13.

```
REM View to sort SQL into GOOD and GARBAGE
REM
CREATE OR REPLACE VIEW sql_garbage AS
SELECT
  b.username users,
  SUM(a.sharable_mem+a.persistent_mem) Garbage,
  TO_NUMBER(null) good
FROM
   sys.v_$sqlarea a, dba_users b
WHERE
  (a.parsing_user_id = b.user_id and a.executions<=1)
GROUP BY b.username
UNION
SELECT DISTINCT
  b.username users,
  TO_NUMBER(null) garbage,
  SUM(c.sharable_mem+c.persistent_mem) Good
FROM
   dba_users b, sys.v_$sqlarea c
WHERE
  (b.user_id=c.parsing_user_id and c.executions>1)
GROUP BY b.username;
```

Source 13: Example Script to Create the SQL_GARBAGE View

Note high lighted area in code, alter this to make the view more restrictive.

```
column garbage format a14 heading 'Non-Shared SQL'
column good format a14 heading 'Shared SQL'
column good_percent format a14 heading 'Percent Shared'
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
column users format a14 heading Users
column nopr noprint
set feedback off
@title80 'Shared Pool Utilization'
spool rep_out\&db\sql_garbage
select 1 nopr,
a.users users,
to_char(a.garbage,'9,999,999,999') garbage,
to_char(b.good,'9,999,999,999') good,
to_char((b.good/(b.good+a.garbage))*100,'9,999,999.999')
good_percent
from sql_garbage a, sql_garbage b
where a.users=b.users
and a.garbage is not null and b.good is not null
union
select 2 nopr,
'-------------' users,'--------------' garbage,'-----------
---' good,
'--------------' good_percent from dual
union
select 3 nopr,
to_char(count(a.users)) users,
to_char(sum(a.garbage),'9,999,999,999') garbage,
to_char(sum(b.good),'9,999,999,999') good,
to_char(((sum(b.good)/(sum(b.good)+sum(a.garbage)))*100),'9
,999,999.999') good_percent
from sql_garbage a, sql_garbage b
where a.users=b.users
and a.garbage is not null and b.good is not null
order by 1,3 desc
/
spool off
```

Source 14: Example Report Script For SQL Reuse Statistics

The report in Source 14 shows at a glance (well, maybe a long glance for a system with hundreds of users) which users aren't making good use of reusable SQL. An example report output is shown in Listing 18.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
Date: 01/28/00
Page:   1
Time: 10:45 AM            Shared Pool Utilization
MAULT
                                dw database

Users           Non-Shared SQL Shared SQL     Percent Shared
--------------- -------------- -------------- --------------
JSMITH             371,387,006      1,007,366           .271
NETSPO              10,603,456        659,999          5.860
DCHUN                6,363,158        151,141          2.320
DSSUSER              5,363,057        824,865         13.330
MRCHDXD              4,305,330        600,824         12.246
DWPROC               2,690,086      4,901,400         64.564
CWOOD                  946,199        239,604         20.206
TMANCEOR               877,644         93,323          9.611
GCMATCH                604,369      1,637,788         73.045
MAULT                  445,566      3,737,984         89.350
PRECISE                205,564     46,342,150         99.558
BKUEHNE                154,754         35,858         18.812
SYS                    146,811      9,420,434         98.465
SMANN                  102,460      8,523,746         98.812
MRCHPHP                 56,954         59,069         50.911
MRCHAEM                 42,465         65,017         60.491
--------------  -------------- -------------- --------------
16                 404,553,888     78,358,468         16.226
```

Listing 18: Example Report From Showing SQL Reuse Statistics

Notice in Listing 18 that the JSMITH user only shows 0.271% shared SQL use based on memory footprints. From the report in Listing 18 we would expect a low reuse value for JSMITH.

Through looking at the performance profiles for several large database systems I have found that once the number of SQL areas being managed by the SGA exceeds several thousand (>7-10 thousand) performance begins to degrade. Therefore knowing the total number of SQL areas and how they are dispersed among the users is also an important metric. In a later lesson we will see the hash value report which will give this data. An example run of the

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

hash report for the database being used here for examples gives the result shown in Listing 19.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
Date: 01/28/00
Page:   1
Time: 10:48 AM            Shared Hash Value Report
MAULT                                       dw database

                            Total Hash  SQL With SQL
USERNAME                   Values Same Hash Sharing Hash
-------------------------- ------ --------- -----------
JSMITH                      11681         0     100.000
SYS                          3469         0     100.000
PRECISE                       322         0     100.000
DWPROC                        245         0     100.000
DSSUSER                       125         0     100.000
SMANN                         125         0     100.000
NETSPO                         98         0     100.000
DCHUN                          50         0     100.000
MAULT                          45         0     100.000
GCMATCH                        37         0     100.000
MRCHDXD                        26         0     100.000
DWSYBASE                       21         0     100.000
TMANCEOR                       15         0     100.000
CWOOD                          11         0     100.000
DATAWHSE                        6         0     100.000
SYSTEM                          6         0     100.000
PATROL                          6         0     100.000
MRCHAEM                         5         0     100.000
BKUEHNE                         4         0     100.000
JHUGHES                         2         0     100.000
MRCHPHP                         2         0     100.000
MRCHPWR                         2         0     100.000
WPEREZ                          2         0     100.000
TTIERNEY                        2         0     100.000
BUYRMRA                         1         0     100.000
INVCMTB                         1         0     100.000
JOTOOLE                         1         0     100.000
SHOLETZ                         1         0     100.000
SSCOTT                          1         0     100.000
DEMERY                          1         0     100.000
DBSNMP                          1         0     100.000
                           ------ ---------
sum                         16314         0
```

Listing 19: Example Hash Report showing number of assigned SQL Areas

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Notice how the JSMITH user, as we would expect, is hogging a majority of the SQL areas and causing us to easily exceed the performance barrier for number of SQL areas.

Another useful thing to know is what statements are being repeated. By generating a report which shows the duplicate statements you can then go back to the developers or the users that are generating the statements and help them to correct the statements so they process data more effectively in a reusable manner. The report in Source 15 shows a method for obtaining this information.

```
REM script similar_sql
REM Mike Ault DBMentors International Jan 1999
SET LINES 132 PAGES 55
COL num_of_times HEADING 'Number|Of|Repeats'
COL SUBSTR(a.sql_text,1,60) HEADING 'SubString - 90
Characters'
COL username FORMAT A10 HEADING 'User'
@title132 'Similar SQL'
SPOOL rep_out\&db\similar_sql
SELECT
    b.username,
    substr(a.sql_text,1,90),
    count(a.sql_text) num_of_times
FROM
    v$sqlarea a, dba_users b
WHERE
    a.parsing_user_id=b.user_id
GROUP BY
    b.username,SUBSTR(a.sql_text,1,90)
HAVING
    COUNT(a.sql_text)>=20
ORDER BY 3 DESC
/
SPOOL OFF
CLEAR COLUMNS
TTITLE OFF
```

Source 15: Example report for duplicate SQL

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

An example output from the report in Source 15 is shown in Listing 20.

```
Date: 01/18/00                                                    Page:   1
Time: 09:10 AM                  Similar SQL                 MAULT
                              dw database

                                                                  Number
                                                                     Of
SubString - 90 Characters                                        Repeats
------------------------------------------------------------ ---------
select a.TitleStatus,b.ProductLine,a.Category from UserArea.      4960
select sum(a.OHQty + a.OOQty) from UserArea.InventoryPP a,us      4885
select b.OrderNum, b.OrderLineNum, b.VendorID, c.ProductShor       676
declare  lERROR_ROW DATAWHSE.ERROR_LOG%ROWTYPE;  lDBCallOK n       660
declare  lLLogSeqNum number;  lERROR_ROW DATAWHSE.ERROR_LOG%       660
declare     LPLogSeqNum number;     lERROR_ROW DATAWHSE.ERRO       630
declare  lDBCallOK number := 0;  lFatalError number := 16;         615
SELECT COUNT(*)   FROM DATAWHSE.LOADER_LOG  WHERE DATACOLLEC       571
Select  Sum(TotalSales + Estimate),sum(TAStoreSales.FieldPla       199
select logging from sys.dba_tablespaces where tablespace_nam       194
SELECT COMMENTS FROM ALL_COL_COMMENTS WHERE TABLE_NAME = 'PO        82
Select  TAStore.DistrictNo,Sum(TotalSales + Estimate),sum(TA        82
SELECT COMMENTS FROM ALL_COL_COMMENTS WHERE TABLE_NAME = 'CO        65
declare  lDBCallOK number := 0;  begin   :lParmResult := lDB        64
Select  TAStore.RegionNo,Sum(TotalSales + Estimate),sum(TASt        56
Select (LastName || ' ' || FirstName) From UserArea.WaldensB        51
Select Subject From UserArea.FFCTitleCTCurrent where SKU = '        51
Select ProductLongName From UserArea.FFCTitleCTCurrent where        51
Select AuthorName From UserArea.FFCTitleCTCurrent where SKU         51
Select BuyerName From UserArea.TitleCurrent Where Bkey in (S        51
Select FFCBuyerCode From   UserArea.FFCTitleInvMgmt where SK        51
Select  TAStore.StoreNo,Sum(TotalSales + Estimate),sum(TASto        46
select A.status, A.timestamp, A.last_ddl_time from all_objec        36
SELECT COMMENTS FROM ALL_COL_COMMENTS WHERE TABLE_NAME = 'DW        35
SELECT COMMENTS FROM ALL_COL_COMMENTS WHERE TABLE_NAME = 'AS        33
Select  sum(TAStoreSales.NewSignUps),sum(TAStoreSales.Renewa        33
Select  Sum(TYSales+Est),sum(TABrdsSales.Forecast),Sum(TYSal        33
SELECT status, timestamp, last_ddl_time FROM  all_objects  W        32
Select  sum(TATitleSMmM.SaleAmt),sum(TATitleSMmM.SaleQty) Fr        30
declare  lLogRow DATAWHSE.LOADER_LOG%ROWTYPE;  lERROR_ROW DA        27
Select  TATitleSMmM.BKey,TATitleSMmM.BKey,TATitleSMmM.TitleN        23
select b.OrderNum, c.ProductShortName, a.LastName, a.FirstNa        22
SELECT A.TRANSEQNUM TRANSEQNUM,A.JRNLSEQNUM JRNLSEQNUM,A.JRN        20
SELECT COMMENTS FROM ALL_COL_COMMENTS WHERE TABLE_NAME = 'BO        20
Select  TATitleSMmM.BKey,TATitleSMmM.TitleName,TATitleSMmM.A        20
```

Listing 20: Example Listing from Similar SQL Report

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Usually you won't be able to add the user name to the report in Listing 20, on a lark I decided to see what would happen if I tied the statements back to the users in the test database. Sure enough JSMITH owned the first two sets of duplicate statements.

Putting it All In Perspective

So what have we seen so far? We have examined reports that show both gross and detailed shared pool usage and whether or not shared areas are being reused. What can we do with this data? Ideally we will use the results to size our shared pool properly. Let's set out a few general guidelines for shared pool sizing:

- **Guideline 1**: If gross usage of the shared pool in a non-ad-hoc environment exceeds 95% (rises to 95% or greater and stays there) establish a shared pool size large enough to hold the fixed size portions, pin reusable packages and procedures. Increase shared pool by 20% increments until usage drops below 90% on the average.
- **Guideline 2**: If the shared pool shows a mixed ad-hoc and reuse environment establish a shared pool size large enough to hold the fixed size portions, pin reusable packages and establish a comfort level above this required level of pool fill. Establish a routine flush cycle to filter non-reusable code from the pool.
- **Guideline 3**: If the shared pool shows that no reusable SQL is being used establish a shared pool large enough to hold the fixed size portions plus a few megabytes (usually not more than 40) and allow the shared pool modified least recently used (LRU) algorithm to manage the pool.

In guidelines 1, 2 and 3, start at around 40 megabytes for a standard size system. Notice in guideline 2 it is stated that a routine flush cycle should be instituted. This flies in the face of what Oracle Support pushes in their shared pool white papers, however, they

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

work from the assumption that proper SQL is being generated and you want to reuse the SQL present in the shared pool. In a mixed environment where there is a mixture of reusable and non-reusable SQL the non-reusable SQL will act as a drag against the other SQL (I call this shared pool thrashing) unless it is periodically removed by flushing. Source 16 shows a PL/SQL package that can be used by the DBMS_JOB job queues to periodically flush the shared pool only when it exceeds a specified percent full.

```
PROCEDURE flush_it(p_free IN NUMBER) IS
--
CURSOR get_share IS
 SELECT
  SUM(a.bytes)
 FROM
  v$sgastat a
 WHERE
  a.pool = 'shared pool' AND
  a.name <> 'free memory';
--
CURSOR get_var IS
 SELECT
  value
 FROM
  v$parameter
 WHERE
   name = 'shared_pool_size';
--
CURSOR get_time IS
 SELECT
  sysdate
 FROM
  dual;
--
  todays_date     DATE;
  mem_ratio       NUMBER;
  share_mem       NUMBER;
  variable_mem NUMBER;
  cur             INTEGER;
  sql_com   VARCHAR2(60);
  row_proc  NUMBER;
--
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
BEGIN
 OPEN get_share;
 OPEN get_var;
 FETCH get_share INTO share_mem;
 DBMS_OUTPUT.PUT_LINE('share_mem: '||TO_CHAR(share_mem));
 FETCH get_var INTO variable_mem;
 DBMS_OUTPUT.PUT_LINE('variable_mem:
'||TO_CHAR(variable_mem));
 mem_ratio:=share_mem/variable_mem;
 DBMS_OUTPUT.PUT_LINE(TO_CHAR(mem_ratio,'99.999')||
 ' '||TO_CHAR(p_free/100,'99.999'));
 IF mem_ratio>p_free/100 THEN
  cur:=DBMS_SQL.OPEN_CURSOR;
  sql_com:='ALTER SYSTEM FLUSH SHARED_POOL';
  DBMS_SQL.PARSE(cur,sql_com,dbms_sql.v7);
  row_proc:=DBMS_SQL.EXECUTE(cur);
  DBMS_SQL.CLOSE_CURSOR(cur);
  OPEN get_time;
  FETCH get_time INTO todays_date;
  INSERT INTO dba_running_stats VALUES (
   'Flush of Shared Pool',mem_ratio,35,todays_date,0);
  COMMIT;
 END IF;
END flush_it;
```

Source 16: Example Script to Create a Shared Pool Flush Routine

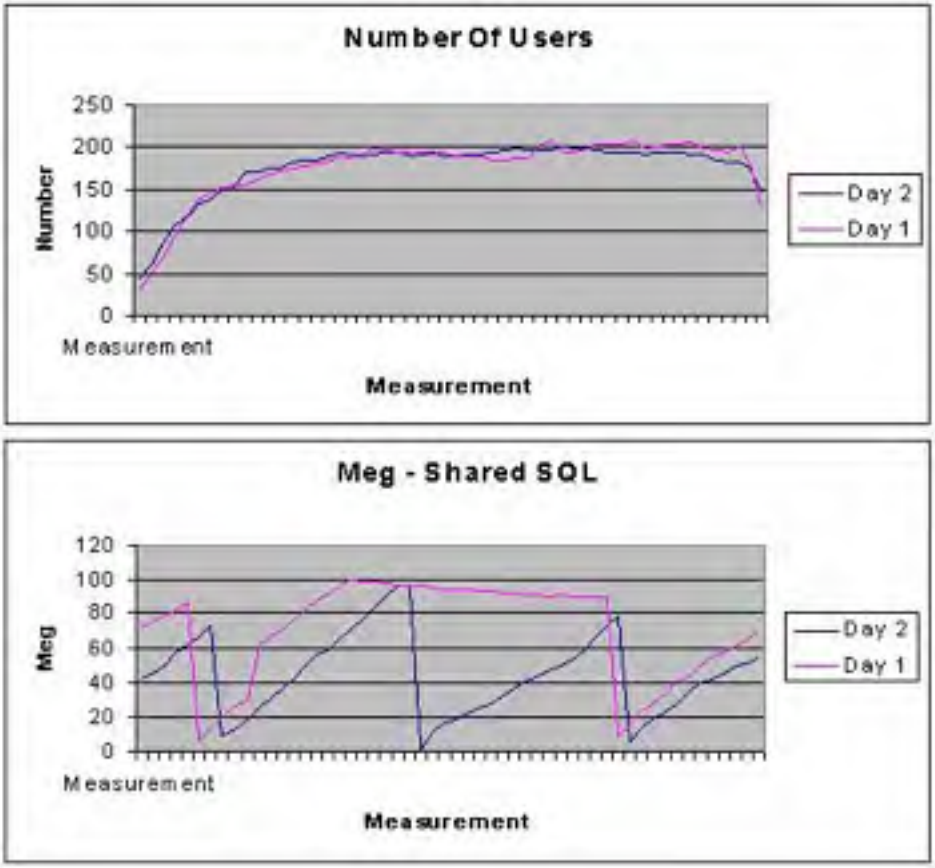
The command set to perform a flush on a once every 30 minute cycle when the pool reaches 95% full would be:

```
VARIABLE x NUMBER;
BEGIN
 dbms_job.submit(
  :X,'BEGIN flush_it(95);
END;',SYSDATE,'SYSDATE+(30/1440)');
END;
/
COMMIT;
(Always commit after assigning a job or the job will not be
run and queued)
```

There is always a discussion as to whether this really does help performance so I set up a test on a production instance where on

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

day 1 I did no automated flushing and on day 2 I instituted the automated flushing. Figure 7 shows the graphs of performance indicators and users.



COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

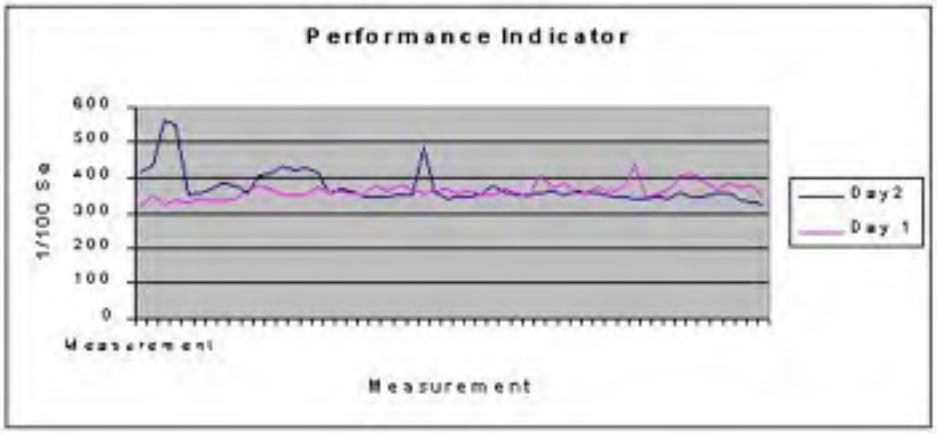


Figure 7: Graphs Showing Effects of Flushing

The thing to notice about the graphs in figure 7 is the overall trend of the performance indicator between day 1 and day 2. On day 1 (the day with an initial flush as indicated by the steep plunge on the pool utilization graph followed by the buildup to maximum and the flattening of the graph) the performance indicator shows an upward trend. The performance indicator is a measure of how long the database takes to do a specific set of tasks (from the Q Diagnostic tool from Savant Corporation). Therefore an increase in the performance indicator indicates a net decrease in performance. On day 2 the overall trend is downward with the average value less than the average value from day 1. Overall the flushing improved the performance as indicated by the performance indicator by 10 to 20 percent. Depending on the environment I have seen improvements of up to 40-50 percent. At a recent job site the shared pool was sized at 210 megabytes and was filled to 170 megabytes. The response time for an internally stored complex PL/SQL routine (the running_stats procedure) was 30 minutes. I flushed the shared pool and response time dropped to a little over a minute.

One thing that made the analysis difficult was that on day 2 there were several large batch jobs run which weren't run on day 1. The

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

results still show that flushing has a positive effect on performance when the database is a mixed SQL environment with a large percentage of non-reusable SQL areas.

Guideline 3 also brings up an interesting point, you may already have over allocated the shared pool, and in this case guideline 3 may result in you decreasing the size of the shared pool. In this situation the shared pool has become a cesspool filled with nothing but garbage SQL. After allocating enough memory for dictionary objects and other fixed areas and ensuring that the standard packages and such are pinned, you should only maintain a few megabytes above and beyond this level of memory for SQL statements. Since none of the code is being reused you want to reduce the hash search overhead as much as possible, you do this by reducing the size of the available SQL area memory so as few a number of statements are kept as possible.

What to Pin

In all of the guidelines stated so far I mention that the memory is usually allocated above and beyond that needed for fixed size areas and pinned objects. How do you determine what to pin? Generally speaking any package, procedure, function or cursor that is frequently used by your application should be pinned into the shared pool when the database is started. I suggest adding a “null” startup function to every in house generated package it essentially looks like Source 17.

```
FUNCTION start_up RETURN number IS
  Ret NUMBER:=1;
BEGIN
  Ret:=0
  RETURN ret;
END start_up;
```

Source 17: Example Null Startup Function

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The purpose of the null startup function is to provide a touch point to pull the entire package into the shared pool. This allows you to create a startup SQL procedure that pulls all of the application packages into the pool and pins them using the DBMS_SHARED_POOL package. The DBMS_SHARED_POOL package may have to be built in earlier releases of Oracle. The DBMS_SHARED_POOL package is built using the DBMSPOOL.SQL and PRVTPOOL.PLB scripts located in (UNIX) $ORACLE_HOME/rdbms/admin or (NT) x:\orant\rdbms\admin (where x: is the home drive for your install).

How do you determine what packages, procedures of functions to pin? Actually, Oracle has made this easy by providing the V$DB_OBJECT_CACHE view that shows all objects in the pool, and, more importantly, how they are being utilized. The script in figure 15.32 provides a list of objects that have been loaded more than once and have executions greater than one. Some example output from this script is shown in Source 18. A rule of thumb is that if an object is being frequently executed and frequently reloaded it should be pinned into the shared pool.

```
rem
rem FUNCTION: Report On Objects Which Should Be Kept
rem
COLUMN owner FORMAT A11 HEADING Schema
COLUMN name FORMAT A22 HEADING 'Should Keep|Object Name'
COLUMN HEADING Name|Space
COLUMN type FORMAT A20 HEADING Object|Type
COLUMN kept FORMAT A4 HEADING Kept
COLUMN sharable_mem FORMAT 999,999 HEADING Shared|Memory
COLUMN executions FORMAT 9,999,999 HEADING Executes
SET LINES 132 PAGES 47 FEEDBACK OFF
@title132 'Oracle Should Keep Report'
BREAK ON owner ON namespace ON type
SPOOL rep_out/&db/should_keeps
SELECT
      owner,
      namespace,
      type,
      name,
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
      sharable_mem,
      loads,
      executions,
      locks,
      pins,
      kept
FROM
      v$db_object_cache
WHERE
      type NOT IN ('NOT LOADED','NON-
EXISTENT','VIEW','TABLE',
                  'SEQUENCE','INVALID TYPE')
      AND loads>1 AND executions>loads AND executions>100
AND kept='NO'
ORDER BY owner,namespace,type,executions desc;
SPOOL OFF
SET LINES 80 PAGES 22 FEEDBACK ON
CLEAR COLUMNS
CLEAR BREAKS
TTITLE OFF
```

Source 18: Script to Show Objects Which Should Be Kept

The output from the script in Source 18 is shown in Listing 21. Notice the objects with high executions.

Listing 21. Example Output From the Script In Source 18.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Date: 12/16/99 Page: 1
Time: 12:54 PM Oracle Should Keep Report SYSTEM
SBSFAPRD database

Schema	Name Space	Object Type	Should Keep Object Name	Shared Memory	LOADS	Executes	LOCKS	PINS	Kept
SIEBEL	TABLE/PROCEDURE	PROCEDURE	NGS_EXPORT_PROC	63,310	51	5,471	72	0	NO
	TRIGGER	TRIGGER	S_OPTY_ESCL_T1	6,733	130	########	92	0	NO
			S_OPTY_POSTN_ESCL_T2	7,035	635	94,678	41	0	NO
			S_OPTY_PROD_ESCL_T1	6,466	110	78,297	151	0	NO
			S_ACCNT_POSTN_ESCL_T1	6,308	674	40,756	10	0	NO
			S_OPTY_POSTN_ESCL_T3	2,971	60	38,236	1	0	NO
			S_ACCNT_POSTN_ESCL_T2	6,308	674	36,428	12	0	NO
			S_ACCNT_POSTN_ESCL_T3	6,372	220	31,664	4	0	NO
			S_OPTY_PROD_ESCL_T2	6,618	167	24,587	14	0	NO
			S_OPTY_CON_ESCL_T2	6,273	664	17,926	14	0	NO
			S_OPTY_ESCL_T2	6,981	156	16,783	6	0	NO
			S_OPTY_POSTN_ESCL_T1	6,707	705	16,306	25	0	NO
			S_CONTACT_ESCL_T2	6,064	183	11,077	10	0	NO
			S_ORG_EXT_T_ESCL_T2	6,882	467	7,022	8	0	NO
			S_ORG_EXT_ESCL_T2	7,368	448	7,021	8	0	NO
			S_OPTY_CON_ESCL_T3	6,481	792	3,669	5	0	NO
			S_SRV_REQ_ESCL_T1	6,984	678	3,429	6	0	NO
			S_OPTY_PROD_ESCL_T3	6,602	420	2,393	2	0	NO
			S_ADDR_ORG_ESCL_T2	7,473	203	1,987	9	0	NO
			NGS_U_TR	5,743	39	1,515	28	0	NO
			S_ORG_EXT_ESCL_T3	6,016	140	443	2	0	NO
			S_ORG_EXT_T_ESCL_T3	6,162	81	194	2	0	NO
			S_ADDR_ORG_ESCL_T3	6,409	38	114	3	0	NO
			S_ORG_INDUST_ESCL_T3	6,331	33	102	2	0	NO
SYS	BODY	PACKAGE BODY	STANDARD	28,700	11	397,486	170	0	NO
			DBMS_APPLICATION_INFO	4,209	40	164,523	2	0	NO
			DBMS_OUTPUT	9,399	30	134,819	1	0	NO
			UTL_FILE	9,380	53	7,210	73	0	NO
	TABLE/PROCEDURE	PACKAGE	STANDARD	120,844	107	397,486	170	0	NO
			DBMS_APPLICATION_INFO	12,161	278	173,486	2	0	NO
			DBMS_STANDARD	15,097	109	161,973	234	0	NO
			DBMS_OUTPUT	14,615	148	100,434	1	0	NO
			UTL_FILE	3,596	82	7,210	73	0	NO

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Note that you only have to pin the package, not the package and package body.

Another criteria for determining if an object should be pinned into the shared pool is its size. The DBMS_SHARED_POOL.SIZES procedure searches the shared pool for any objects larger than the size in kilobytes of the argument it is passed. Generally, the larger the size the more likelihood that the object is a package and you will want to keep it in the pool, smaller objects tend to be individual queries and can be aged out of the pool. Remember that the DBMS_SHARED_POOL procedure is not generally automatically loaded when an instance is built, the DBMSPOOL.SQL and PRVTPOOL.PLB scripts must be run from INTERNAL or SYS users for it to be created. The use of DBMS_SHARED_POOL.SIZES is shown in Listing 22.

```
SQL> set serveroutput on size 4000;
SQL> execute sys.dbms_shared_pool.sizes(10);
SIZE(K) KEPT   NAME
------- ------ ----------------------------------------------------------
139            SYS.STANDARD            (PACKAGE)
56             SYS.DBMS_SHARED_POOL          (PACKAGE BODY)
31             SELECT TO_CHAR(SHARABLE_MEM / 1000 ,'999999')
SZ,DECODE(KEPT_VE
               RSIONS,0,'      ',RPAD('YES(' || TO_CHAR(KEPT_VERSIONS)  |
               | ')' ,6)) KEEPED,RAWTOHEX(ADDRESS) || ','  || TO_CHAR(HASH
               _VALUE)  NAME,SUBSTR(SQL_TEXT,1,354) EXTRA   FROM V$SQLAREA
               WHERE SHARABLE_MEM > :b1 * 1000   UNION SELECT TO_CHAR(SH
               ARABLE_MEM / 1000 ,'999999') SZ,DECODE(KEPT,'YES','YES
               (004D7F84,2008220828)    (CURSOR)
30             SYS.STANDARD                  (PACKAGE BODY)
27             SYS.DBMS_SHARED_POOL          (PACKAGE)
17             SYS.V$SQLAREA         (VIEW)
16             SYS.V$DB_OBJECT_CACHE  (VIEW)
15             insert into idl_ub2$(obj#,part,version,piece#,length,piece)
val
               ues(:1,:2,:3,:4,:5,:6)
               (0027BA44,-512326869)    (CURSOR)
PL/SQL procedure successfully completed.
```

Listing 22. Example of the use of the DBMS_SHARED_POOL.SIZES procedure

The ‘set serveroutput on size 4000’ command in Listing 22 limits the size of the output buffer to 4000 bytes. The ‘set serveroutput’

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

command is required. Perhaps in the future if we all bug Oracle for an enhancement they will incorporate the use of UTIL_FILE and just generate us a report listing 21 we can review as we desire. As you can see from Listing 22, there is one large package in shared memory. Let's issue a keep against this package to retain it. Listing 23 shows the results from this action.

```
SQL> execute dbms_shared_pool.keep('sys.standard');
PL/SQL procedure successfully completed.
SQL> execute dbms_shared_pool.sizes(130);

SIZE(K) KEPT   NAME
------- ------ ----------------------------------------
139     YES    SYS.STANDARD          (PACKAGE)
PL/SQL procedure successfully completed.
```

Listing 23: Example of the use of DBMS_SHARED_POOL.KEEP packaged procedure.

By issuing keeps against large packages to hold them in memory you can mitigate shared pool fragmentation that results in the ORA-04031 error. By pinning the packages so they don't age out this prevents smaller queries, cursors and procedures from taking their areas and then when the packages are reloaded, viola! an ORA-04031 as the package seeks a large enough group of areas in which to install itself. Under ORACLE8 this is supposed to be eliminated due to the way the shared memory area is now used, however I have had some reports on as late a version as 8.0.5.

Guideline 4: Determine usage patterns of packages, procedures, functions and cursors and pin those that are frequently used.

The Shared Pool and MTS

The use of the multi-threaded server option (MTS) in Oracle requires a sometimes-dramatic increase in the size of the shared pool. This increase in the size of the shared pool caused by MTS is due to the addition of the user global areas required for sorting and

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

message queues. If you are using MTS you should monitor the V$SGASTAT values for MTS related memory areas and adjust the shared pool memory allocations accordingly.

Note that in Oracle 8 you should make use of the large pool feature to pull the user global areas (UGA) and multi-threaded server queues out of the shared pool area if MTS is being used. This prevents the fragmentation problems that have been reported in shared pools when MTS is used without allocating the large pool. The parallel query option (PQO) in Oracle8 also makes use of the large pool area, depending on the number of users and degree of parallel, the large pool may require over 200 megabytes by itself in a PQO environment.

Large Pool Sizing

Sizing the large pool can be complex. The large pool, if configured must be at least 600 kilobytes in size. Usually for most MTS applications 600k is enough. However, if PQO is also used in your Oracle8 environment then the size of the large pool will increase dramatically. The V$SGASTAT dynamic performance view has a new column in Oracle8, POOL. The POOL column in the V$SGASTAT view is used to contain the pool area where that particular type of object is being stored. By issuing a summation select against the V$SGASTAT view a DBA can quickly determine the size of the large pool area currently being used.

```
SELECT name, SUM(bytes) FROM V$SGASTAT WHERE pool='LARGE
POOL'
GROUP BY ROLLUP(name);
```

The above select should be used when an "ORA-04031:Unable to allocate 16084 bytes of shared memory ("large pool", "unknown object", "large pool hea", "PX large pool") " error is received during operation with a large pool configured (the number of bytes specified may differ). When the above select is run, the resulting summary number of bytes will indicate the current size of the pool and show how close you are to your maximum as specified in the

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

initialization parameter LARGE_POOL_SIZE. Generally increasing the large_pool by up to 100% will eliminate the ORA-04031 errors.

Oracle8i provides for automated sizing of the large pool. If PARALLEL_AUTOMATIC_TUNING is set to TRUE or if PARALLEL_MAX_SERVERS is set to a non-zero value then the LARGE_POOL_SIZE will be calculated, however, it can be over-ridden with a manually specified entry in the initialization file. Indeed, if an ORA-27102: Out of Memory error is received when you set either of these parameters (or both) you must either manually set LARGE_POOL_SIZE or reduce the value for PARALLEL_MAX_SERVERS. The following formula determines the set point for the LARGE_POOL_SIZE if it is not manually set:

```
(DOP^2*(4I-1)+2*DOP*3+4*DOP(I-1))*PEMS*USERS
```

Where:

- DOP – Degree of Parallel calculated from #CPU/NODE * #NODES
- I – Number of threads/CPU
- PEMS – Parallel execution message size – set with PARALLEL_EXECUTION_MESSAGE_SIZE initialization parameter, usually defaults to 2k or 4k but can be larger
- USERS – Number of concurrent users using parallel query

For a 2k PEMS with 4 concurrent users for a steadily increasing value for DOP the memory size is a quadratic function ranging from around 4 meg for 10 CPUs to 120 meg with 70 CPUs. This memory requirement is demonstrated in Figure 15.36.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

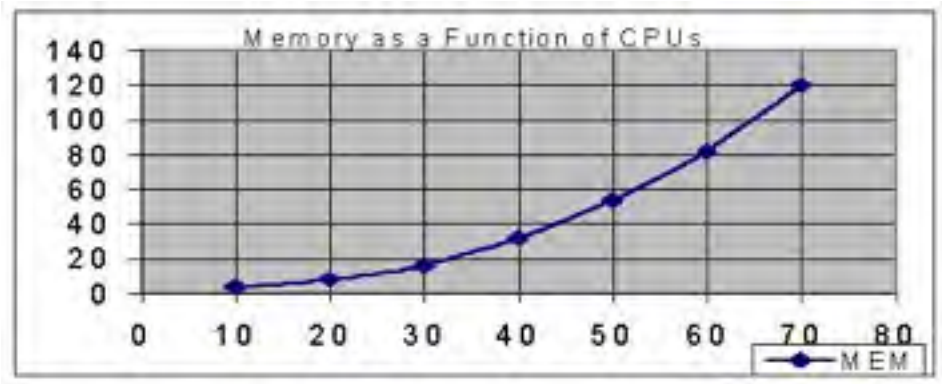


Figure 8: Example Chart for 2k PEMS and 4 Concurrent Users Showing Memory Requirements as Number of CPUs Increases

On my NT4.0 Oracle8i, 8.1.3 test system I have 2 CPUs, set at 2 threads per cpu (DOP of 4) and then 4 threads per CPU (DOP of 8), message buffer of 4k and I performed multiple tests increasing the PARALLEL_MAX_SERVERS initialization parameter to see what the resulting increase in LARGE_POOL_SIZE would be, the results are shown in Table 17.

PARALLEL_MAX_ SERVERS	DOP 4 LARGE_POOL_SIZE	DOP 8 LARGE_POOL_SIZE
4	685,024 bytes	685,024 bytes
8	857,056 bytes	857,056 bytes
16	1,151,968 bytes	1,545,184 bytes

Table 17: Large Pool Size as a Function of Parallel Max Servers and DOP

Notice that for a small number of CPUs the large pool size increase from an increase in parallel max servers isn't affected by changes in the number of parallel threads until the value of threads is large in respect to the number of CPUs.

For non-PQO systems a general rule of thumb is 5K of memory for each MTS user for the large pool area.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Guideline 5: In Oracle7when using MTS increase the shared pool size to accommodate MTS messaging and queuing as well as UGA requirements. In Oracle8 use the Large Pool to prevent MTS from effecting the shared pool areas. If using PQO the large pool is required in Oracle8.

A Matter Of Hashing

We have discussed hashing in prior lessons, essentially each SQL statement is hashed and this hash value is then used to compare to already stored SQL areas, if a matching hash is found the statements are compared. The hash is only calculated based on the first 200 or so characters in the SQL statement, so extremely long SQL statements can result in multiple hashes being the same even though the stored SQL is different (if the first 100 or so characters in each statement are identical). This is another argument for using stored procedures and functions to perform operations and for the use of bind variables. There is hope, in 8i (or 8.1 if you prefer) the hash value will be calculated on the first 100 and last 100 characters reducing the chances of multiple identical hash values for different SQL statements.

If the number of large, nearly identical statements is high, then the number of times the parser has to compare a new SQL statement to existing SQL statements with the same hash value increases. This results in a higher statement overhead and poorer performance. You should identify these large statements and encourage users to re-write them using bind variables or to proceduralize them using PL/SQL. The report in Source 24 will show if you have a problem with multiple statements being hashed to the same value.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
rem: FUNCTION: Shows by user who has possible
rem:           SQL reuse problems
COLUMN total_hash                          HEADING 'Total
Hash|Values'
COLUMN same_hash                     HEADING 'SQL With|Same
Hash'
COLUMN u_hash_ratio     FORMAT 999.999    HEADING 'SQL
Sharing|Hash'
START title80 'Shared Hash Value Report'
SPOOL rep_out\&&db\shared_hash.lst
BREAK ON REPORT
COMPUTE SUM OF total_hash ON REPORT
COMPUTE SUM OF same_hash ON REPORT
SELECT
      a.username,
      count(b.hash_value) total_hash,
      count(b.hash_value)-COUNT(UNIQUE(b.hash_value))
same_hash,
(COUNT(UNIQUE(b.hash_value))/COUNT(b.hash_value))*100
u_hash_ratio
FROM
      dba_users a, v$sqlarea b
WHERE
      a.user_id=b.parsing_user_id
GROUP BY
      a.username;
CLEAR COMPUTES
```

Source 24: Example Script to Report on Hashing Problems

The script in Source 24 produces a report similar to that shown in Listing 23. The report in Listing 23 shows which users are generating SQL that hashes to the same values. Once you have a user isolated you can then run the script in Source 12 to find the bad SQL statements.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
Date: 11/20/98
Page:   1
Time: 11:40 AM            Shared Hash Value Report
AULTM
                              DCARS database
                               Total Hash  SQL With SQL
Sharing
USERNAME                           Values Same Hash
Hash
------------------------- ---------- --------- -----------
AULTM                             129         0     100.000
DCARS                            6484        58      99.105
QDBA                              109         0     100.000
RCAPS                             270         0     100.000
RCOM                              342         7      97.953
SECURITY_ADMIN                     46         0     100.000
SYS                               134         0     100.000
                             ---------- ---------
sum                              7514        65
```

Listing 23: Hash Report

A quick glance at the report in Listing 23 shows that I need to look at the DCARS user to correct the hashing problems they might be having and improve the reuse of SQL in the shared pool. However, look at the number of hash areas this user has accumulated, 6,484, if I run the report from Source 12 in the first half of this lesson it will out weigh the paper version of the Oracle documentation set. A faster way to find the hash values would be to do a self-join and filter out the hash values that are duplicated. Sounds easy enough, but remember, the V$ tables have no ROWIDs so you can't use the classic methods, you have to find another column that will be different when the HASH_VALUE column in V$SQLAREA is the same. Look at the select in Source 25.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
select distinct a.hash_value from v$sqlarea a, v$sqlarea b,
dba_users c
where a.hash_value=b.hash_value and
a.parsing_user_id = c.user_id
and c.username='DCARS' and  ← change to user you are
concerned about
a.FIRST_LOAD_TIME != b.FIRST_LOAD_TIME
```

Source 25: Example Select To Determine Duplicate Hash Values

Listing 24 has an example output from the above select.

```
DCARS:column hash_value format 99999999999
DCARS:set echo on
DCARS: select distinct a.hash_value from v$sqlarea a,
v$sqlarea b,
  2  dba_users c
  3  where a.hash_value=b.hash_value and
  4  a.parsing_user_id = c.user_id
  5  and c.username='DCARS' and
  6* a.FIRST_LOAD_TIME != b.FIRST_LOAD_TIME

  HASH_VALUE
------------
 -1595172473
 -1478772040
 -1344554312
  -941902153
  -807684425
  -507978165
  -270812489
   441376718
   784076104
   979296206
  1765990350
  1945885214
```

Listing 24: Example Hash Select Output

Once you have the hash value you can pull the problem SQL statements from either V$SQLAREA or V$SQLTEXT very easily, look at Listing 25.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
DCARS:select sql_text from v$sqlarea where hash_value='441376718';

SQL_TEXT
----------------------------------------------------------------------
SELECT   region_code,   region_dealer_num,   consolidated_dealer_num,
dealer_name,   dealer_status_code,   dealer_type_code,   mach_credit_code,
parts_credit_code  FROM dealer  WHERE   region_code = '32' AND
reg_dealer_num = '6433'

SELECT   region_code,   region_dealer_num,   consolidated_dealer_num,
dealer_name,   dealer_status_code,   dealer_type_code,   mach_credit_code,
parts_credit_code  FROM dealer  WHERE   region_code = '56' AND
reg_dealer_num = '6273'
```

Listing 25: Example of Statements With Identical Hash Values But Different SQL

Long statements require special care to see that bind variables are used to prevent this problem with hashing. Another help for long statements is to use views to store values at an intermediate state thus reducing the size of the variable portion of the SQL. Notice in the example select in Listing 25 that the only difference between the two identically hashed statements is that the "region_code" and "reg_dealer_num" comparison values are different, if bind variables had been used in these statements there would only have been one hash entry instead of two.

Guideline 6: Use bind variables, PL/SQL (procedures or functions) and views to reduce the size of large SQL statements to prevent hashing problems.

Disk IO and the Shared Pool

The shared SQL area contains the Pcode versions of all of the current SQL commands that haven't been aged out of the shared pool. There are numerous statistics available via the v$sqlarea DPT. The text of SQL statements in the shared pool can be retrieved (at least the first tens of bytes) from the v$sqltext DPT. Lets look at a report that displays the SQL statements in the SQL area with the greatest amount of disk reads (these will probably be

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

the ones you will want to review and tune). Look at the report in Source 26.

```
REM Name: sqldrd.sql
REM Function: return the sql statements from the shared
area with
REM Function: highest disk reads
REM History: Presented in paper 35 at IOUG-A 1997,
converted for
REM use 6/24/97 MRA
REM
DEFINE access_level = 1000 (NUMBER)
COLUMN parsing_user_id FORMAT 9999999 HEADING 'User Id'
COLUMN executions       FORMAT 9999        HEADING 'Exec'
COLUMN sorts            FORMAT 99999       HEADING 'Sorts'
COLUMN command_type     FORMAT 99999       HEADING 'CmdT'
COLUMN disk_reads       FORMAT 999,999,999 HEADING 'Block
Reads'
COLUMN sql_text   FORMAT a40  HEADING 'Statement'
WORD_WRAPPED
SET LINES 130 VERIFY OFF FEEDBACK OFF
START title132 'SQL Statements With High Reads'
SPOOL rep_out/&db/sqldrd.lis
SELECT
      parsing_user_id, executions,
      sorts,command_type,
      disk_reads,sql_text
FROM
      v$sqlarea
WHERE
      disk_reads > &&access_level
ORDER BY
      disk_reads;
SPOOL OFF
SET LINES 80 VERIFY ON FEEDBACK ON
```

Source 26: Script to Monitor SQL Area Disk Reads By Script

```
Date: 06/24/97
Page:   1
Time: 11:35 PM           SQL Statements With High Reads
SYSTEM
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

ORTEST1 database

User Id	Exec	Sorts	CmdT	Block Reads	Statement
0	403	0	3	11	select f.file#, f.block#, f.ts#, f.length from fet$ f, ts$ t where t.ts#=f.ts# and t.dflextpct!=0
0	11	0	3	11	select order#,columns,types from access$ where d_obj#=:1
0	12	0	3	12	select /*+ index(idl_ub1$ i_idl_ub11) +*/ piece#,length,piece from idl_ub1$ where obj#=:1 and part=:2 and version=:3 order by piece#
5	34	0	3	13	SELECT NAME,VALUE FROM V$SYSSTAT WHERE NAME = 'db block gets'
0	12	0	3	14	select /*+ index(idl_ub2$ i_idl_ub21) +*/ piece#,length,piece from idl_ub2$ where obj#=:1 and part=:2 and version=:3 order by piece#
0	17	0	3	27	select file#, block#, ts# from seg$ where type# = 3
0	1	1	3	79	select distinct d.p_obj#,d.p_timestamp from sys.dependency$ d, obj$ o where d.p_obj#>=:1 and d.d_obj#=o.obj# and o.status!=5
5	34	0	47	90	DECLARE job BINARY_INTEGER := :job; next_date DATE := :mydate; broken

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
                        BOOLEAN := FALSE; BEGIN
                        hitratio;
                        :mydate := next_date; IF broken
                        THEN :b := 1; ELSE :b := 0; END
                        IF; END;
```

Listing 26: Example Output From SQL Disk Read Script

The example report in Listing 26 was generated forcing a read count of 10. Usually disk reads will be in the range specified by the define statement. By tuning those statements which show large amounts of disk reads the overall performance of the application is increased.

Monitoring Library and Data Dictionary Caches

I've spent most of this lesson looking at the shared SQL area of the shared pool. Let's wrap up with a high level look at the library and data dictionary caches. The library cache area is monitored via the V$LIBRARYCACHE view and contains the SQL area, PL/SQL area, table, index and cluster cache areas. The data dictionary caches contain cache area for all data dictionary related definitions.

The script in Source 27 creates a report on the library caches. The items of particular interest in the report generated by the script in Source 27 (shown in Lisitng 27) are the various ratios. This report should look familiar, it is similar to the library caches report in the utlestat.sql reports.

```
REM
REM Title: libcache.sql
REM
REM FUNCTION: Generate a library cache report
REM
COLUMN namespace                        HEADING "Library
Object"
COLUMN gets             FORMAT 9,999,999 HEADING "Gets"
COLUMN gethitratio      FORMAT 999.99    HEADING "Get Hit%"
COLUMN pins             FORMAT 9,999,999 HEADING "Pins"
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
COLUMN pinhitratio      FORMAT 999.99    HEADING "Pin Hit%"
COLUMN reloads          FORMAT 99,999    HEADING "Reloads"
COLUMN invalidations    FORMAT 99,999    HEADING  "Invalid"
COLUMN db               FORMAT a10
SET PAGES 58 LINES 80
START title80 "Library Caches Report"
DEFINE output = rep_out\&db\lib_cache
SPOOL &output
SELECT
      namespace,
      gets,
      gethitratio*100 gethitratio,
      pins,
      pinhitratio*100 pinhitratio,
      reloads,
      invalidations
FROM
      v$librarycache
/
SPOOL OFF
PAUSE Press enter to continue
SET PAGES 22 LINES 80
TTITLE OFF
UNDEF output
```

Listing 27: Example Script To Monitor The Library Caches

Look at the example output from the script in Source 27 in Listing 27. In Listing 27 we see that all Get Hit% (gethitratio in the view) except for indexes are greater than 80-90 percent. This is the desired state, the value for indexes is low because of the few accesses of that type of object. Notice that the Pin Hit% is also greater than 90% (except for indexes) this is also to be desired. The other goals of tuning this area are to reduce reloads too as small a value as possible (this is done by proper sizing and pinning) and to reduce invalidations. Invalidations happen when for one reason or another an object becomes unusable. However, if you must use flushing of the shared pool reloads and invalidations may occur as objects are swapped in and out of the shared pool. Proper pinning can reduce the number of objects reloaded and invalidated.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Guideline 7: In a system where there is no flushing increase the shared pool size in 20% increments to reduce reloads and invalidations and increase hit ratios.

```
Date: 11/21/98
Page:   1
Time: 02:51 PM             Library Caches Report
SYSTEM
                                ORTEST1 database
                         Get           Pin
Library Object     Gets Hit%  Pins    Hit%  Reloads Invalid
--------------- ------ ----- ------ ----- ------- -------
SQL AREA        46,044 99.17 99,139 99.36      24      16
TABLE/PROCEDURE  1,824 84.59  6,935 93.21       3       0
BODY               166 93.98    171 91.23       0       0
INDEX               27   .00     27   .00       0       0
CLUSTER            373 98.12    373 97.59       0       0
```

Listing 27: Example Of The Output From Library Caches Report

The data dictionary caches used to be individually tunable through several initialization parameters, now they are internally controlled. The script in Source 28 should be used to monitor the overall hit ratio for the data dictionary caches.

```
REM
REM title:    ddcache.sql
REM FUNCTION: report on the v$rowcache table
REM HISTORY:  created sept 1995 MRA
REM
START title80 "DD Cache Hit Ratio"
SPOOL rep_out\&db\ddcache
SELECT (SUM(getmisses)/SUM(gets)) ratio
FROM v$rowcache
/
SPOOL OFF
PAUSE Press enter to continue
TTITLE OFF
```

Source 28: Script to Monitor the Data Dictionary Caches

The output from the script in Source 28 is shown in Listing 28.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
Date: 11/21/98
Page:   1
Time: 02:59 PM                DD Cache Hit Ratio
SYSTEM
                               ORTEST1 database

    RATIO
---------
.01273172
```

Listing 28: Example Output From Data Dictionary Script

The ratio reported from the script in Source 28 should always be less than 1. The ratio corresponds to the number of times out of 100 that the database engine sought something from the cache and missed. A dictionary cache miss is more expensive than a data block buffer miss so if your ratio gets near 1 increase the size of the shared pool since the internal algorithm isn't allocating enough memory to the data dictionary caches.

Guideline 8: In any shared pool, if the overall data dictionary cache miss ratio exceeds 1 percent, increase the size of the shared pool.

In Summary

I hope you now understand that the old "just increase the shared pool" answer isn't good enough anymore when it comes to tuning problems. You must take an in depth look at your shared pool and tune what needs to be tuned, not just throw memory at a problem until it submerges. Indeed, I have shown that in some cases increasing the size of the shared pool may harm performance and decreasing the size may be advisable. The shared pool is vital to the proper performance of your Oracle database, you must have it properly tuned or drown in bad performance. Next we will cover what to pin, the shared pool and multi-threaded server, hashing and generalized library and dictionary cache tuning. We have also discussed ways to monitor for what objects should be pinned, discussed multi-threaded server, looked at hashing problems and

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

their resolution as well as examined classic library and data dictionary cache tuning. We have established 8 guidelines for tuning the Oracle shared pool:

Guideline 1: If gross usage of the shared pool in a non-ad-hoc environment exceeds 95% (rises to 95% or greater and stays there) establish a shared pool size large enough to hold the fixed size portions, pin reusable packages and procedures. Gradually increase shared pool by 20% increments until usage drops below 90% on the average.

Guideline 2: If the shared pool shows a mixed ad-hoc and reuse environment, establish a shared pool size large enough to hold the fixed size portions, pin reusable packages and establish a comfort level above this required level of pool fill. Establish a routine flush cycle to filter non-reusable code from the pool.

Guideline 3: If the shared pool shows that no reusable SQL is being used establish a shared pool large enough to hold the fixed size portions plus a few megabytes (usually not more than 40) and allow the shared pool modified least recently used (LRU) algorithm to manage the pool. (also see guideline 8)

Guideline 4: Determine usage patterns of packages, procedures, functions and cursors and pin those that are frequently used.

Guideline 5: In Oracle7when using MTS increase the shared pool size to accommodate MTS messaging and queuing as well as UGA requirements. In Oracle8 use the Large Pool to prevent MTS from effecting the shared pool areas.

Guideline 6: Use bind variables, PL/SQL (procedures or functions) and views to reduce the size of large SQL statements to prevent hashing problems.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Guideline 7: In a system where there is no flushing increase the shared pool size in 20% increments to reduce reloads and invalidations and increase object cache hit ratios.

Guideline 8: In any shared pool, if the overall data dictionary cache miss ratio exceeds 1 percent, increase the size of the shared pool.

Using these guidelines and the scripts and techniques covered in this lesson, your should be well on the way towards a well tuned and well performing shared pool.

Table 18: Initialization Parameters That Effect The Shared Pool

NAME	DESCRIPTION
shared_pool_size	size in bytes of shared pool (7 and 8)
shared_pool_reserved_size	size in bytes of reserved area of shared pool (7 and 8)
shared_pool_reserved_min_alloc	minimum allocation size in bytes for reserved area of shared pool (7 and 8)
large_pool_size	size in bytes of the large allocation pool (8 only)
parallel_max_servers	Maximum number of parallel query slaves, if set forces calculation and setting of large pool size parameter (8i)
parallel_adaptive_multi_user	If set forces calculation of large pool size is over-ridden if size manually set(8i)

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	DESCRIPTION
parallel_automatic_tuning	If set forces calculation of large pool size is over-ridden if size is manually set (8i)
large_pool_min_alloc	minimum allocation size in bytes for the large allocation pool (8 only, obsolete in 8i)
parallel_min_message_pool	minimum size of shared pool memory to reserve for pq servers (8 only, obsolete in 8i)
backup_io_slaves	Number of backup IO slaves to configure (8 only)
temporary_table_locks	Number of temporary table locks to configure (7 and 8)
dml_locks	Number of DML locks to configure (7 and 8)
sequence_cache_entries	Number of sequence numbers to cache (7 and 8)
row_cache_cursors	Number of row caches to set up (7 and 8)
max_enabled_roles	Number of role caches to set up (7 and 8)
mts_dispatchers	Number of MTS dispatcher processes to start with (7 and 8)
mts_max_dispatchers	Maximum number of dispatcher processes to allow (7 and 8)
mts_servers	Number of MTS servers to start with (7 and 8)
mts_max_servers	Maximum number of MTS servers to allow (7 and 8)

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

NAME	DESCRIPTION
open_cursors	Maximum number of open cursors per session (7 and 8)
Cursor_space_for_time	Hold open cursors until process exits (7 and 8)

Table 19: Initialization Parameters Used In Tuning Shared Pool

View Name	Purpose
V$PARAMETER	Contains current settings for all documented initialization parameters
V$SGASTAT	Contains sizing information for all SGA areas
V$SQLAREA	Contains information and statistics on the SQL area of the shared pool
V$DB_OBJECT_CACHE	Contains information on all cached objects in the database shared pool area
V$LIBRARYCACHE	Contains statistics on the library caches
V$ROWCACHE	Contains statistics on the data dictionary caches
DBA_USERS	Contains database user information
V$BUFFER_POOL	Oracle8 view showing pool areas
V$BUFFER_POOL_STATISTICS	Oracle8 buffer pool statistics
V$BH	View that monitors every buffer in buffer pool

Table 20: Views Dealing With Shared Pool and Buffer Tuning

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Software	Manufacturer	Purpose
Oracle Administrator	RevealNet, Inc.	Administration Knowledge base
Q Diagnostic	Savant, Corp.	Provide Oracle DB diagnostics

Table 21: Software Mentioned in Lessons

Managing the Database

There are many tasks that a DBA must perform when managing the database. Among the most important (besides backup and recovery which we already covered) is the ability to resolve lock and deadlock situations. A lock situation is when one user is holding a lock on a resource when another user requires it. A lock situation usually involves one of two things, either a poorly written application or a user who has not committed a transaction. Both of these lock situations are resolved by finding the problem session or user and either picking up the phone and getting the user themselves to correct the situation or killing the users session via database commands.

Dead-lock situations occur when users require the same resource in such a manner that neither that can release it without the other releasing first. Usually Oracle will resolve deadlocks by rolling back the transaction of the first user process that detects the deadlock.

Find USER locking others/Kill problem USER

In order to find a lock holding user I suggest that the DBA be proactive and create the DBA_WAITERS and DBA_BLOCKERS views using the catblock.sql script which is located in the $ORACLE_HOME/rdbms/admin directory or its equivalent. Once the waiter and blocker views are created a DBA can simply query

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

either view to find the lock situation. Sources 29 and 30 show scripts to query these views.

```
rem NAME: blockers.sql
rem FUNCTION: Show all processes causing a dead lock
rem HISTORY: MRA 1/15/96 Created
rem
COLUMN username         FORMAT a10  HEADING 'Holding|User'
COLUMN session_id                       HEADING 'SID'
COLUMN mode_held        FORMAT a20  HEADING 'Mode|Held'
COLUMN mode_requested   FORMAT a20  HEADING 'Mode|Requested'
COLUMN lock_id1         FORMAT a20  HEADING 'Lock|ID1'
COLUMN lock_id2         FORMAT a20  HEADING 'Lock|ID2'
COLUMN type                             HEADING
'Lock|Type'
SET LINES 132 PAGES 59 FEEDBACK OFF ECHO OFF
START title132 'Sessions Blocking Other Sessions Report'
SPOOL rep_out\&db\blockers
SELECT
      a.session_id,
      username,
      type,
      mode_held,
      mode_requested,
      lock_id1,
      lock_id2
FROM
      sys.v_$session b,
      sys.dba_blockers c,
      sys.dba_lock a
WHERE
      c.holding_session=a.session_id and
      c.holding_session=b.sid
/
SPOOL OFF
PAUSE press enter/return to continue
CLEAR COLUMNS
SET LINES 80 PAGES 22 FEEDBACK ON ECHO ON
```

Source 29: Example script to report on blocking locks

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
rem
rem FUNCTION: Report on sessions waiting for locks
rem
COLUMN busername FORMAT A10 HEADING 'Holding|User'
COLUMN wusername FORMAT A10 HEADING 'Waiting|User'
COLUMN bsession_id HEADING 'Holding|Sid'
COLUMN wsession_id HEADING 'Waiting|Sid'
COLUMN mode_held FORMAT A20 HEADING 'Mode|Held'
COLUMN mode_requested FORMAT A20 HEADING 'Mode|Requested'
COLUMN lock_id1 FORMAT A20 HEADING 'Lock|Id1'
COLUMN lock_id2 FORMAT A20 HEADING 'Lock|Id2'
COLUMN type HEADING 'LOCK|TYPE'
SET LINES 132 PAGES 59 FEEDBACK OFF ECHO OFF
START TITLE132 'Processes Waiting on Locks Report'
SPOOL rep_out/&db/waiters
SELECT
      holding_session bsession_id,
      waiting_session wsession_id,
      b.username busername,
      a.username wusername,
      c.lock_type type,
      mode_held, mode_requested,
      lock_id1, lock_id2
FROM
sys.v_$session b, sys.dba_waiters c, sys.v_$session a
WHERE
c.holding_session=b.sid and
c.waiting_session=a.sid
/
SPOOL OFF
PAUSE press enter/return to continue
CLEAR COLUMNS
SET LINES 80 PAGES 22 FEEDBACK ON ECHO ON
TTITLE OFF
```

Source 30: Example script to report on blocking and waiting sessions.

Example output from the blockers report is shown in Listing 29 for a simple lock situation.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
Date: 02/08/00
Page:   1
Time: 07:01 AM                Sessions Blocking Other Sessions Report
SYSTEM
                                        aultdb database

          Holding    Lock       Mode         Mode                 Lock
Lock
      SID User       Type       Held         Requested            ID1
ID2
--------- ---------- ---------- ------------ -------------------- --------
------------ ------------
        9 LABUSER    USER       Row-X (SX)   None                 2821
0
        9 LABUSER    USER       Exclusive    None                 196702
5547
```

Listing 29: example Blockers Report

The output for the same lock situation from the waiters report is shown in Listing 30.

```
Date: 02/08/00
Page:   1
Time: 07:01 AM                                     Processes Waiting on
Locks Report                                     SYSTEM
                                                          aultdb database

  Holding   Waiting Holding    Waiting     Lock         Mode
Mode                Lock                  Lock
      SID       SID User       User        Type         Held
Requested           ID1                   ID2
--------- --------- ---------- ----------- ------------ -----------------
-- -------------------- -------------------- ------------
        9        13 LABUSER    LABUSER2   Transaction   Exclusive
Exclusive           196702                5547
```

Listing 30: Example Waiters report

While both reports show who is holding the lock, I prefer the waiters report since it also shows who is being blocked . Using the reports it becomes a simple matter to retrieve the required information from the V$SESSION view to either notify the locking process owner or simply kill the offending process outright if it is a non-critical process. Tools such as Precise*SQL and Q from Savant provide

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

lock status and the Q product allows killing of processes in a fairly easy manner.

Methods of Murder

What are the methods of killing these recalcitrant sessions? Essentially, you can issue an ALTER SYSTEM KILL SESSION or you can issue a manual process kill command such as the UNIX 'kill -9 pid' from the operating system side. You should do one or the other of these types of kill operation, but not both. If you kill both the Oracle process and the operating system process it can result in database hang situations where you will have to perform a shutdown abort. If you know the PID and user name it is trivial to select the required information from V$SESSION, for example from the above reports:

```
SELECT sid, serial# from V$SESSION WHERE sid=9 AND
username='LABUSER';
```

Killing From the Oracle Side

The DBA can either manually issue an ALTER SYSTEM command or develop a dynamic SQL script to perform the operation for him or her. Source 31 shows a PL/SQL procedure to perform a kill of a process using the dynamic SQL package of procedures; DBMS_SQL. In Oracle8i a new option is available for the ALTER SYSTEM which allows disconnection of the user process after a transaction completes. The remarked section in Source 31 shows this alternative to a straight kill.

```
CREATE OR REPLACE PROCEDURE kill_session ( session_id in
varchar2,
serial_num in varchar2)
AS
cur INTEGER;
ret INTEGER;
string VARCHAR2(100);
BEGIN
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
 string :=
        'ALTER SYSTEM KILL SESSION' || CHR(10) ||
CHR(39)||session_id||','||serial_num||CHR(39);
-- string :=
--          'ALTER SYSTEM DISCONNECT SESSION' || CHR(10) ||
-- CHR(39)||session_id||','||serial_num||CHR(39)||CHR(10)||
--' POST_TRANSACTION';
   cur := dbms_sql.open_cursor;
   dbms_sql.parse(cur,string,dbms_sql.v7);
   ret := dbms_sql.execute(cur)  ;
   dbms_sql.close_cursor(cur);
EXCEPTION
   WHEN OTHERS THEN
      raise_application_error(-20001,'Error in
execution',TRUE);
      IF dbms_sql.is_open(cur) THEN
        dbms_sql.close_cursor(cur);
      END IF;
END;
/
```

Source 31: Example execution of the kill_session script

```
EXECUTE kill_session('13','1424');
```

Once a session is killed, its status in the V$SESSION view goes to "KILLED" and users will receive an error if they try to reactivate the session. The session entry will not be removed until the user attempts to reconnect. Shutdown immediate and shutdown normal are supposed to be able to handle killed sessions properly but there have been some reports of problems up to version 7.3.2 on some platforms.

For the above locking situation the resulting actions would look like:

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
SQL> SELECT sid, serial# from V$SESSION WHERE sid=9 AND
username='LABUSER';

      SID   SERIAL#
--------- ---------
        9       171

1 row selected.


SQL> execute kill_session(9,171);

PL/SQL procedure successfully completed.

SQL> @waiters
press enter/return to continue
```

Killing From the Operating System Side

The other method of removing these unwanted sessions is to kill them from the operating system side. In UNIX environments this is accomplished with the “kill -9’ command executed from a privileged user. In other operating systems there are similar commands. Before a session can be killed from the operating side you must determine which process belongs to the user. In a MTS environment this may be impossible and thus the Oracle side kill method is preferred in MTS situations.

```
kill -9 11240
```

Source 32: Example kill command from UNIX

Creating and starting the database

Other important DBA responsibilities are the creation of databases and the startup and shutdown of databases.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Database Creation

To create a database, the CREATE command is run under SVRMGR or with 8i, in SQL*Plus.

1. First, the DBA must connect to the Oracle internal user (or SYS user in 8i) via the command:

```
CONNECT INTERNAL  AS SYSDBA
```

2. Next, the instance is started in an unmounted condition. This is accomplished with the following command.

```
STARTUP NOMOUNT PFILE=filename
```

   where:

   PFILE=filename refers to the database initialization file (INIT.ORA) you will be using, unless it is located in the directory you are currently in; a path must also be provided.

3. Next, the database is created. The format would be:

```
CREATE DATABASE name
   CONTROLFILE REUSE
   LOGFILE GROUP n (filespec)
   MAXLOGFILES n
   MAXLOGMEMBERS     n
   MAXLOGHISTORY     n
   MAXDATAFILES      n
   MAXINSTANCES      n
   ARCHIVELOG|NOARCHIVELOG
   CHARACTER_SET  charset
   NATIONAL_CHARACTER_SET charset
   DATAFILE (filespec) autoextend_clause
     Extent_management_clause

autoextend_clause:
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
AUTOEXTEND ON|OFF NEXT n K|M MAXSIZE n|UNLIMITED K|M

Filespec:

'full path file name|logical|system link name' SIZE n
K|M REUSE

extent_management_clause:

EXTENT MANAGEMENT
              DIRECTORY
              LOCAL
                AUTOALLOCATE
                UNIFORM
                      SIZE n [K|M]
```

where:

Database name is the name of the database, maximum of eight characters long.

File specifications for data files are of the format: ‘filename’ SIZE integer K or M REUSE. K is for kilobytes, M is for Megabytes. REUSE specifies that if the file already exists, reuse it. New with later versions of Oracle7 and all of Oracle8 is the AUTOEXTEND option, which is used to allow your datafiles to automatically extend as needed. Be very careful with this command as it can use up a great deal of disk space rather rapidly if a mistake is made during table builds or inserts.

File specifications for log files depend on the operating system.

The MAXLOGFILES, MAXDATAFILES, and MAXINSTANCES set hard limits for the database; these should be set to the maximum you ever expect.

MAXLOGMEMBERS and MAXLOGHISTORY are hard limits.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

EXTENT MANAGEMENT – This determines if the extents in the SYSTEM tablespace are managed via the data dictionary (DICTIONARY) or locally via a bitmap in the tablespace (LOCAL). In addition the extents can be AUTOALLOCATED letting the system manage them as to size or the UNIFORM clause with or without a size specification can be used to force all extents to a uniform size.

For Oracle7 an Oracle8, CHARACTER_SET and NATIONAL_CHARACTER_SET determine the character set that data will be stored in; this value is operating system dependent.

If you need archive logging, set ARCHIVELOG; if you don't need it right away, set NOARCHIVELOG. I suggest using NOARCHIVELOG to start to avoid creation of multiple archive logs during database initial creation, you won't recover from a failed build, you will just rebuild. This is one thing to check if your latter steps of the build (running catproc.sql for example) seem to stall, the archive log location may have filled. This is checked using the alert log stored in the location specified by user_dump_destination.

Databases are created in EXCLUSIVE mode. Databases are either EXCLUSIVE or PARALLEL. A database must be altered to PARALLEL mode after creation.

The CHARACTER_SET is for normal data. Character specifications JA16SJISFIXED, JA16EUCFIXED and JA16DBCSFIXED can only be used as the NATIONAL_CHARACTER_SET. The NATIONAL_CHARACTER_SET specifies the national character set used to store data in columns specifically defined as NCHAR, NCLOB, or NVARCHAR2. You cannot change the national character set after creating the database. If not specified, the national character set defaults to the database character set.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

You must verify that the registry on NT is set with the same character set as the database or data problems and performance degradation will occur. This applies to all client workstations including Windows95, 97 and 98 as well as Windows2000 (NT5) based units.

What the system does when given a CREATE DATABASE command is easy. First, the system creates control, redo log, and database files. Next, the system creates the SYSTEM rollback segment in the SYSTEM tablespace, creates and loads data dictionary tables, and mounts and opens the database.

On NT and other Windows based installs the instance manager will help with creation of new instances. To access it use the `START` button and then choose the `PROGRAMS` menu item. From the list of `PROGRAMS` choose the `Oracle for Windows NT` folder. Inside the `Oracle for Windows NT` folder will be an icon for `NT Instance Manager v8.0` (or whatever version you are running). Selection of the `NT Instance manager v8.0` icon will start the instance manager, which will list the current (if any) instances of Oracle running on your platform. Select the `New` button from this display. You will be prompted for all required information to start a base installation of Oracle. The major limitation of this tool is that it only provides single line inputs for items that have multiple lines of values. However, be patient; it can be used once you are used to the way it works. Another painful part of its use is that if you make a mistake, it forgets everything you told it and you have to start over. The newer versions are more reliable.

Re-creation of a Database

Periodically it may be required that a DBA recreate a database. Perhaps the block size was incorrectly specified, perhaps it has to be moved from one location to another, or perhaps a DBA has inherited a system and just wants to see how the database was created. I wish I could say there was an easy way to get this

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

information from the database, but unfortunately unless you have some of the third-party tools, it just isn't so.

Hopefully, you are with a shop that has detailed information on the hows, whens, and whys databases were created. Unfortunately, most shops seem to do this in the SOTP mode (seat of the pants) so no actual documentation is available. The lesson to be learned here is to always use a script to create anything in the database that is permanent. Source 32 shows an example script to document a database create statement for an Oracle instance. Not documented in this script are the MAX set of parameters, the reason being that these are stored in the control file. To completely document the CREATE command, also document the control file with:

```
ALTER DATABASE BACKUP CONTROL FILE TO TRACE;
```

The file will be located in the background_dump_destination location specified in the v$parameter table. An example output from the ALTER DATABASE BACKUP CONTROLFILE TO TRACE is shown in Listing 33. Another item which must be documented is the initialization parameters for your database; these are located in the v$parameter virtual table. The script in Source 33 documents these for you in an almost-ready-for-prime-time format.

```
REM FUNCTION: SCRIPT FOR CREATING 8.1 DB FROM EXISTING DB
REM           This script must be run by a user with the
DBA role.
REM           This script is intended to run with 8i.
REM           Running this script will in turn create a
script to
REM           rebuild the database.  This created
REM           script, crt_db.sql,  is run by SVRMGR or
SQLPLUS
REM         Only preliminary testing of this script was
performed.
REM           Be sure to test it completely before relying
on it.
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
REM M. Ault 2/19/99 DMR Consulting, REVELNET
REM
SET VERIFY OFF FEEDBACK OFF ECHO OFF PAGES 0
SET TERMOUT ON
PROMPT Creating db build script...
SET TERMOUT OFF;
REM
REM Create Holding Table
REM
CREATE TABLE db_temp
     (lineno NUMBER,  text VARCHAR2(255))
/
REM
REM Procedure
REM
DECLARE
--
-- Cursor Declarations
--
CURSOR get_block IS
     SELECT value
         FROM v$parameter
          WHERE name='db_block_size';
--
CURSOR ts_cursor IS
     SELECT   initial_extent,
              next_extent,
              min_extents,
              max_extents,
              pct_increase,
              min_extlen,
              extent_management,
              allocation_type
            FROM    sys.dba_tablespaces
            WHERE tablespace_name = 'SYSTEM';
--
CURSOR df_cursor IS
     SELECT    file_name,
               bytes,
               autoextensible,
               maxbytes,
               increment_by
      FROM     sys.dba_data_files
      WHERE    tablespace_name = 'SYSTEM'
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
      ORDER BY file_name;
--
CURSOR grp_cursor IS
     SELECT group#
     FROM   v$log;
--
CURSOR mem_cursor (grp_num number) IS
     SELECT   a.member,
              b.bytes
     FROM     v$logfile a,
              v$log b
     WHERE    a.group#=grp_num
          AND a.group#=b.group#
     ORDER BY
          member;
--
-- Variable Declarations
--
   block_size           NUMBER;
   lv_initial_extent
sys.dba_tablespaces.initial_extent%TYPE;
   lv_next_extent
sys.dba_tablespaces.next_extent%TYPE;
   lv_min_extents
sys.dba_tablespaces.min_extents%TYPE;
   lv_max_extents
sys.dba_tablespaces.max_extents%TYPE;
   lv_pct_increase
sys.dba_tablespaces.pct_increase%TYPE;
   lv_file_name         sys.dba_data_files.file_name%TYPE;
   lv_bytes             sys.dba_data_files.bytes%TYPE;
   lv_max_extend        sys.dba_data_files.maxbytes%TYPE;
   lv_ext_incr
sys.dba_data_files.increment_by%TYPE;
   lv_autoext
sys.dba_data_files.autoextensible%TYPE;
   lv_first_rec         BOOLEAN;
   lv_min_extlen
sys.dba_tablespaces.min_extlen%TYPE;
   lv_extent_man
sys.dba_tablespaces.extent_management%TYPE;
   lv_allocation
sys.dba_tablespaces.allocation_type%TYPE;
   sub_strg             VARCHAR2(20);
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
   grp_member           v$logfile.member%TYPE;
   bytes                v$log.bytes%TYPE;
   db_name              VARCHAR2(8);
   db_string            VARCHAR2(255);
   db_lineno            NUMBER := 0;
   thrd                 NUMBER;
   grp                  NUMBER;
   sz                   NUMBER;
   begin_count          NUMBER;
   max_group            NUMBER;
--
-- Local use procedures
--
PROCEDURE write_out(p_line INTEGER,
                 p_string VARCHAR2) IS
   BEGIN
    INSERT INTO db_temp (lineno,text)
    VALUES (p_line,p_string);
   END;
--
-- Start of actual code
--
BEGIN
   SELECT MAX(group#) INTO max_group FROM v$log;
     db_lineno:=db_lineno+1;
   SELECT 'CREATE DATABASE '||name INTO db_string
     FROM v$database;
     write_out(db_lineno,db_string);
     db_lineno:=db_lineno+1;
   SELECT 'CONTROLFILE REUSE' INTO db_string
     FROM dual;
     write_out(db_lineno,db_string);
     db_lineno:=db_lineno+1;
   SELECT 'LOGFILE ' INTO db_string
     FROM dual;
     write_out(db_lineno,db_string);
COMMIT;
IF grp_cursor%ISOPEN
THEN
     CLOSE grp_cursor;
     OPEN grp_cursor;
ELSE
     OPEN grp_cursor;
END IF;
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
LOOP
     FETCH grp_cursor INTO grp;
     EXIT WHEN grp_cursor%NOTFOUND;
     db_lineno:=db_lineno+1;
     db_string:= ' GROUP '||grp||' (';
     write_out(db_lineno,db_string);
     IF mem_cursor%ISOPEN THEN
          CLOSE mem_cursor;
          OPEN mem_cursor(grp);
     ELSE
          OPEN mem_cursor(grp);
     END IF;
     db_lineno:=db_lineno+1;
     begin_count:=db_lineno;
     LOOP
          FETCH mem_cursor INTO grp_member, bytes;
          EXIT when mem_cursor%NOTFOUND;
          IF begin_count=db_lineno THEN
               db_string:=chr(39)||grp_member||chr(39);
               write_out(db_lineno,db_string);
               db_lineno:=db_lineno+1;
          ELSE
               db_string:='
,'||chr(39)||grp_member||chr(39);
               write_out(db_lineno,db_string);
               db_lineno:=db_lineno+1;
          END IF;
     END LOOP;
     db_lineno:=db_lineno+1;
     IF grp=max_group THEN
          db_string:=' ) SIZE '||bytes;
          write_out(db_lineno,db_string);
     ELSE
          db_string:=' ) SIZE '||bytes||',';
          write_out(db_lineno,db_string);
     END IF;
END LOOP;
OPEN get_block;
  FETCH get_block INTO block_size;
CLOSE get_block;
OPEN ts_cursor;
  FETCH ts_cursor INTO Lv_initial_extent,
              Lv_next_extent,
              Lv_min_extents,
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
              Lv_max_extents,
              Lv_pct_increase,
              Lv_min_extlen,
              Lv_extent_man,
              Lv_allocation;
lv_first_rec := TRUE;
CLOSE ts_cursor;
OPEN df_cursor;
      LOOP
         FETCH df_cursor INTO lv_file_name,
                              lv_bytes,
                              lv_autoext,
                              lv_max_extend,
                              lv_ext_incr;
         EXIT WHEN df_cursor%NOTFOUND;
         IF (lv_first_rec) THEN
            lv_first_rec := FALSE;
            db_string := 'DATAFILE ';
         ELSE
            db_string := db_string ||chr(10)|| ',';
         END IF;

db_string:=db_string||chr(39)||lv_file_name||chr(39)||
                    ' SIZE '||to_char(lv_bytes) || '
REUSE';
         IF lv_autoext='YES' THEN
          IF lv_max_extend=0 THEN
               sub_strg:=' MAXSIZE UNLIMITED';
          ELSE
               sub_strg:=' MAXSIZE
'||TO_CHAR(lv_max_extend);
          END IF;
          IF lv_ext_incr != 0 THEN
           db_string:=db_string||chr(10)||' AUTOEXTEND ON
NEXT '||
           to_char(lv_ext_incr*block_size)||sub_strg;
          END IF;
         END IF;
         IF lv_min_extlen != 0 AND lv_extent_man!='LOCAL'
THEN
           db_string:=db_string||chr(10)||
          'MINIMUM EXTENT '||TO_CHAR(lv_min_extlen);
         END IF;
      END LOOP;
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
      CLOSE df_cursor;
         db_lineno := db_lineno + 1;
         write_out(db_lineno, db_string);
         db_lineno := db_lineno + 1;
      IF lv_extent_man!='LOCAL' THEN
         db_string := (' DEFAULT STORAGE (INITIAL ' ||
                      TO_CHAR(lv_initial_extent) ||
                      ' NEXT ' || lv_next_extent);
      ELSE
         db_string:=('EXTENT MANAGEMENT '||lv_extent_man);
       IF lv_allocation='UNIFORM' THEN
         db_string:=db_string||' '||lv_extent_man||' SIZE '
         ||lv_initial_extent;
       ELSE
         db_string:=db_string||' AUTOALLOCATE';
       END IF;
      END IF;
         write_out(db_lineno, db_string);
         db_lineno := db_lineno + 1;
      IF lv_extent_man!='LOCAL' THEN
         db_string := (' MINEXTENTS ' ||
                      lv_min_extents ||
                      ' MAXEXTENTS ' || lv_max_extents);
         write_out(db_lineno, db_string);
         db_lineno := db_lineno + 1;
         db_string := (' PCTINCREASE ' ||
                      lv_pct_increase || ')');
         write_out(db_lineno, db_string);
      END IF;
COMMIT;
SELECT
DECODE(value,'TRUE','ARCHIVELOG','FALSE','NOARCHIVELOG')
     INTO db_string FROM v$parameter WHERE
name='log_archive_start';
     db_lineno:=db_lineno+1;
     write_out(db_lineno,db_string);
SELECT ';' INTO db_string from  dual;
     db_lineno:=db_lineno+1;
     write_out(db_lineno,db_string);
CLOSE mem_cursor;
CLOSE grp_cursor;
COMMIT;
END;
/
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
rem The next section could be converted to use
rem UTLFILE so the entire anonymous PL/SQL section
rem and this report section would become a stored
rem procedure, but to keep it generic I will leave as
rem is.
COLUMN dbname NEW_VALUE db NOPRINT
SELECT name dbname FROM v$database;
SET HEADING OFF PAGES 0 VERIFY OFF RECSEP OFF
SPOOL rep_out\&db\crt_db.sql
COLUMN text FORMAT a80 WORD_WRAP
SELECT text
FROM db_temp
ORDER BY lineno;
SPOOL OFF
SET FEEDBACK ON VERIFY ON TERMOUT ON
DROP TABLE db_temp;
PROMPT Press enter to continue
SET VERIFY ON FEEDBACK ON PAGES 22 TERMOUT ON
CLEAR COLUMNS
```

Source 32: Script to recreate the database CREATE command for an instance.

```
REM
REM NAME        : init_ora_rct.sql
REM FUNCTION    : Recreate the instance init.ora file
REM USE         : GENERAL
REM Limitations : None
REM
SET NEWPAGE 0 VERIFY OFF
SET ECHO OFF FEEDBACK OFF TERMOUT OFF PAGES 300 LINES 80
HEADING OFF
COLUMN name  FORMAT a80 WORD_WRAPPED
COLUMN dbname NEW_VALUE db NOPRINT
SELECT name dbname FROM v$database;
DEFINE OUTPUT = 'rep_out\&db\init.ora'
DEFINE cr = chr(10)
SPOOL &OUTPUT
SELECT '# Init.ora file from v$parameter'||&&cr||
'# generated on:'||sysdate||&&cr||
'# script by MRA 11/7/95 REVEALNET'||&&cr||
'#' name FROM dual
UNION
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
SELECT name||' = '||value name  FROM V$PARAMETER
WHERE value IS NOT NULL;
SPOOL OFF
CLEAR COLUMNS
SET NEWPAGE 0 VERIFY OFF
SET TERMOUT ON PAGES 22 LINES 80 HEADING ON
SET TERMOUT ON
UNDEF OUTPUT
PAUSE Press enter to continue
```

Source 33: Script to generate a listing of initialization parameters for your database.

```
Dump file H:\ORAWIN\RDBMS71\trace\ORA14071.TRC
Sat Mar 30 10:05:53 1996
ORACLE V7.1.4.1.0 - Production.
vsnsta=0
vsnsql=a vsnxtr=3
MS-WINDOWS Version 3.10
Sat Mar 30 10:05:52 1996
Sat Mar 30 10:05:53 1996

*** SESSION ID:(5.3)
# The following commands will create a new control file and
use it
# to open the database.
# No data other than log history will be lost. Additional
logs may
# be required for media recovery of offline data files. Use
this
# only if the current version of all online logs are
available.
STARTUP NOMOUNT
CREATE CONTROLFILE REUSE DATABASE "ORACLE" NORESETLOGS
NOARCHIVELOG
    MAXLOGFILES 32
    MAXLOGMEMBERS 2
    MAXDATAFILES 32
    MAXINSTANCES 16
    MAXLOGHISTORY 1600
LOGFILE
  GROUP 1 'H:\ORAWIN\DBS\wdblog1.ora'  SIZE 500K,
  GROUP 2 'H:\ORAWIN\DBS\wdblog2.ora'  SIZE 500K
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
DATAFILE
  'H:\ORAWIN\DBS\wdbsys.ora' SIZE 10M,
  'H:\ORAWIN\DBS\wdbuser.ora' SIZE 3M,
  'H:\ORAWIN\DBS\wdbrbs.ora' SIZE 3M,
  'H:\ORAWIN\DBS\wdbtemp.ora' SIZE 2M
;

# Recovery is required if any of the datafiles are restored
backups,
# or if the last shutdown was not normal or immediate.
RECOVER DATABASE

# Database can now be opened normally.
ALTER DATABASE OPEN;
```

Listing 32: Example output from the ALTER DATABASE BACKUP CONTROL FILE command.

Database Startup and Shutdown

When the instance and database are created, on UNIX the oratab file should have the instance name added to it. This allows the operating system command files in the $ORACLE_HOME/ bin/dbstart and $ORACLE_HOME/bin/dbshut files to see the instance, these files startup and shutdown the databases listed in the oratab .

On NT4.0, startup and shutdown are generally handled by the strt<SID>.cmd and orashut.bat files located in (assuming your home drive is C:) C:/orant/database. The oradim7x (where x is the subversion) program is used to start, stop, and maintain the databases on NT4.0. By specifying command sets in the .CMD files different actions can be taken in regard to the Oracle database system. For example, the startdb.cmd file for an instance with a sid of TEST, an INTERNAL password of ORACLE, might look like this:

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
c:\orant\bin\oradim73 -startup -sid TEST -usrpwd ORACLE
                      -pfile
c:\oracle1\ortest1\admin\pfile\initORTEST1.ora
                      -starttype SRVC, INST
```

A shutdown script for the same instance would look like this:

```
c:\orant\bin\oradim73      -shutdown -sid TEST
                           -SURPWD ORACLE
                           -SHTTYPE  SRVC, INST
                           -SHUTMODE a
```

Startup

The database is open and ready for use after being created. Once the operating system is shut down, or the database is shut down, it must be started before it can be accessed.

UNIX Startup. On UNIX systems, the DBA has to perform the following steps to ensure the instance and database startup each time the system starts up.

1. Log in as root.
2. Edit the /etc/oratab file. Change the last field for your $ORACLE_SID to Y.
3. Add a line similar to the following to your /etc/rc file; be sure you use the full path to the dbstart procedure.

```
su - oracle_owner -c /users/oracle/bin/dbstart
```

Manual Startup. On all systems manual startup is accomplished either via the supplied scripts or through the SVRMGR or in later Oracle8i release, SQLPLUS program. To startup a database using SVRMGR or SQLPLUS, use the following procedure. The command used is the STARTUP command; its format follows.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
STARTUP  [RESTRICTED] [FORCE] [PFILE=filename]
          [EXCLUSIVE or PARALLEL]
          [MOUNT or OPEN] dbname
          [NOMOUNT]
          [RECOVER]
```

1. Log in to SVRMGR as INTERNAL or into SQLPLUS as SYS.
2. Issue one of the following commands:
   a) STARTUP OPEN dbname PFILE=filename This command starts the instance, and opens the database named dbname using the parameter file specified by the filename following the PFILE= clause. This starts up the database in the default, EXCLUSIVE mode.
   b) STARTUP RESTRICT OPEN dbname PFILE=filename This command starts the instance, and opens the database named dbname using the parameter file specified by the filename following the PFILE= clause. This starts up the database in the restricted only mode (only users with RESTRICTED SESSION privilege can log in).
   c) STARTUP NOMOUNT This command starts the instance, but leaves the database dismounted and closed. Cannot be used with EXCLUSIVE, MOUNT, or OPEN. Normally this command is used only when creating a database. There are some maintenance activities that require the database to be in NOMOUNT but generally it is only used with database creation.
   d) STARTUP MOUNT This command starts the instance and mounts the database, but leaves it closed.
   e) STARTUP OPEN dbname PARALLEL
   f) This command starts the instance, opens the database and puts the database in PARALLEL mode for multi-instance use in pre-Oracle8 versions. In Oracle8, simply setting the initialization parameter PARALLEL_SERVER to TRUE starts the instance in parallel server (shared) mode. PARALLEL is

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

obsolete in Oracle8. It cannot be used with EXCLUSIVE or NOMOUNT or if the INIT.ORA parameter SINGLE_PROCESS is set to TRUE. The SHARED parameter is obsolete in Oracle8.

3. STARTUP OPEN dbname EXCLUSIVE

   This command is functionally identical to (a) above. Cannot be specified if PARALLEL or NOMOUNT is also specified in pre-Oracle8 versions. EXCLUSIVE is obsolete in Oracle8. If PARALLEL_SERVE is FALSE, the database defaults to EXCLUSIVE.

4. The FORCE parameter can be used with any of the above options to force a shutdown and restart of the database into that mode. This is not normally done and is only used for debugging and testing.
5. The RECOVER option can be used to immediately start recovery of the database on startup if desired.

Errors that can occur during a startup include missing files, improperly specified PFILE path or name, or corrupted file errors. If these occur, the database will immediately shut down. Using OEM (Oracle Enterprise Manager) you must log in as an account that has been assigned the SYSOPER or SYSDBA roles in order to start up or shut down an instance.

Shutdown

The databases should be shut down before system shutdowns, before full backups and any time system operations require it to be shut down.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

UNIX Shutdown. For UNIX there are several things that need to be done to ensure shutdown occurs. The following procedure, for the HP-UX version of UNIX, demonstrates these steps.

1. Log in as root.
2. Edit the /etc/oratab file. Make the last field a Y for the $ORACLE_SID you want shut down.
3. Add the following entry to your /etc/shutdown file. Be sure to use full path to the dbshut utility.

```
su - oracle_owner -c /usr/oracle/bin/dbshut
```

You should alter the shutdown scripts to do a SHUTDOWN IMMEDIATE. This backs out any uncommitted user transactions, logs them out and then shuts down the database. If a normal SHUTDOWN is performed, the system politely waits for all users to log off of Oracle . . . if Joe is on vacation and left his terminal up in a form, you could have a long wait. The other shutdown, SHUTDOWN ABORT, should only be used for emergencies as it stops the database just as it is, with operations pending or not. A SHUTDOWN ABORT will require a recovery on startup.

The above provides for automatic shutdown when the operating system shuts down. For a normal shutdown, execute the dbshut procedure for UNIX, the SHUTDOWN_ sid.COM procedure for VMS, where the sid is the SID for the database you want to shut down. If it has been created, the stop<sid>.cmd script is used to shutdown an Oracle instance on NT.

To perform a manual shutdown on either system, perform the following procedure.

1. Log in to SVRMGR as INTERNAL.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

2. Issue the appropriate SHUTDOWN command.
3. No option means SHUTDOWN NORMAL. The database waits for all users to disconnect, prohibits new connects, then closes and dismounts the database, then shuts down the instance.
4. SHUTDOWN IMMEDIATE. Cancels current calls like a system interrupt, and closes and dismounts the database, then shuts down the instance. PMON gracefully shuts down the user processes. No instance recovery is required on startup.
5. SHUTDOWN ABORT. This doesn't wait for anything. It shuts the database down now. Instance recovery will probably be required on startup. You should escalate to this by trying the other shutdowns first.

Tuning Responsibilities

Since tuning will be covered in other presentations I will just list the major tuning steps and discuss in general the goals for each step. The major thing to remember is that running is recursive, changes to one area may affect others, an example would be the correction of I/O waits resulting in a CPU bottleneck. Before beginning a tuning effort always set goals, what do you want to accomplish? Otherwise you won't know when you are successful.

Step 1: Tune the Business Rules

In this step the goals should be to create a set of business rules that truly model the business needs for the database. The business rules should specify requirements not give solutions or methods.

Step 2: Tune the Data Design

In this step the goals should be to provide a data design that optimizes the access of data. Usually the data structure will be normalized according to the rules of normalization to at least a third normal form design. Once a normal design is reached, specific

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

tables will be denormalized to optimize performance. At this step indexing is laid out to optimize join paths.

Step 3: Tune the Application Design

In this step the DBA must have input to the developers and provide them with tools and guidance in the proper methods to optimize the SQL and PL/SQL used to access data. The goal of this step is to provide a properly tuned and optimized application that utilizes the data design to provide maximum performance.

Step 4: Tune the Logical Structure of the Database

This step may require changes to the data design and additional creation of indexes. In this step the application is analyzed to set such parameters as INITRANS and FREELISTS. Proper setup of sequences is also accomplished in this step.

Step 5: Tune Database Operations

System designers and application developers must understand Oracle's query processing mechanism to write effective SQL statements.

Before tuning the Oracle Server itself, be certain that your application is taking full advantage of the SQL language and the Oracle features designed to speed application processing. Use features and techniques such as the following based on the needs of your application:

- Array processing
- The Oracle optimizer
- The row-level lock manager
- PL/SQL

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Step 6: Tune the Access Paths

Ensure that there is efficient access to data. Consider the use of clusters, hash clusters, B*-tree indexes and bitmap indexes as well as new features of Oracle8 such as index only tables, partitioned views and indexes and using new object oriented features.

Ensuring efficient access may mean adding indexes, or adding indexes for a particular application section (such as for specialized reporting) and then dropping them again. It may mean revisiting your design after you have built the database. You may want to do more denormalization or create alternative indexes at this point. Upon testing the application you may find that you're still not obtaining the required response time. Look for more ways to improve the design.

Step 7: Tune Memory Allocation

Appropriate allocation of memory resources to Oracle memory structures can have a large impact on performance.

Oracle8 shared memory is allocated dynamically to the following structures, which are all part of the shared pool. Although you explicitly set the total amount of memory available in the shared pool, the system dynamically sets the size of each structure contained within it:

- The data dictionary cache
- The library cache
- Context areas (if running a multithreaded server)

You can explicitly set memory allocation for the following structures:

- Buffer cache
- Log buffer

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- Sequence caches

Proper allocation of memory resources can improve cache performance, reduce parsing of SQL statements, and reduce paging and swapping.

Process local areas include:

- Context areas (for systems not running a multithreaded server)
- Sort areas
- Hash areas

Be careful not to allocate to the system global area (SGA) to such a large percentage of the machine's physical memory that it causes paging or swapping.

Step 8: Tune I/O and Physical Structure

Disk I/O tends to reduce the performance of many software applications. Oracle Server, however, is designed so that its performance need not be unduly limited by I/O. Tuning I/O and physical structure involves these procedures:

1. Distributing data so that I/O is distributed, thus avoiding disk contention
2. Storing data in data blocks for best access: setting the right number of free lists, and proper values for PCTFREE and PCTUSED
3. Creating extents large enough for your data so as to avoid dynamic extension of tables, which would hurt high-volume OLTP applications
4. Evaluating the use of raw devices

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Step 9: Tune Resource Contention

Concurrent processing by multiple Oracle users may create contention for Oracle resources. Contention may cause processes to wait until resources are available. Take care to reduce the following kinds of contention:

- Block contention
- Shared pool contention
- Lock contention
- Pinging (in a parallel server environment)
- Latch contention

Step 10: Tune the Underlying Platform(s)

See your platform-specific Oracle documentation to investigate ways of tuning the underlying system. For example, on UNIX-based systems you might want to tune the following:

- Size of the UNIX buffer cache
- UNIX memory segments
- Logical volume managers
- Memory and size for each process

Tuning Summary

Remember that without clear tuning goals you will never know if you have been successful. Tuning is a recursive process and is never finished.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Layout & Fragmentation

As was discussed in the section on OFA, layout of a set of tablespaces can have dramatic affects on performance and maintainability of a database. Internal layout of tablespaces can also cause performance problems if not handled properly. Internal layout of a tablespace deals with table, index, and rollback segment sizing and placement. Once the main OFA rules are enforced, you must further review design to prevent overloading of tablespaces.

Overloading of tablespaces is when many hi-activity segments are placed in a single tablespace. Many third-party vendors that build on top of Oracle are often guilty of overloading. In an ideal setup each table, index and rollback segment would have its own disk or array with an independent IO channel and the CPUs would be able to talk to each area simultaneously. However we all know this is not possible at least with current technology. Overloading results in what is known as hotspotting, this is where a single disk or array becomes saturated with IO requests because of the volume of hot tables or indexes that reside there.

Figure 9 shows the results from Hotspotting through a Precise*SQL screenshot.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

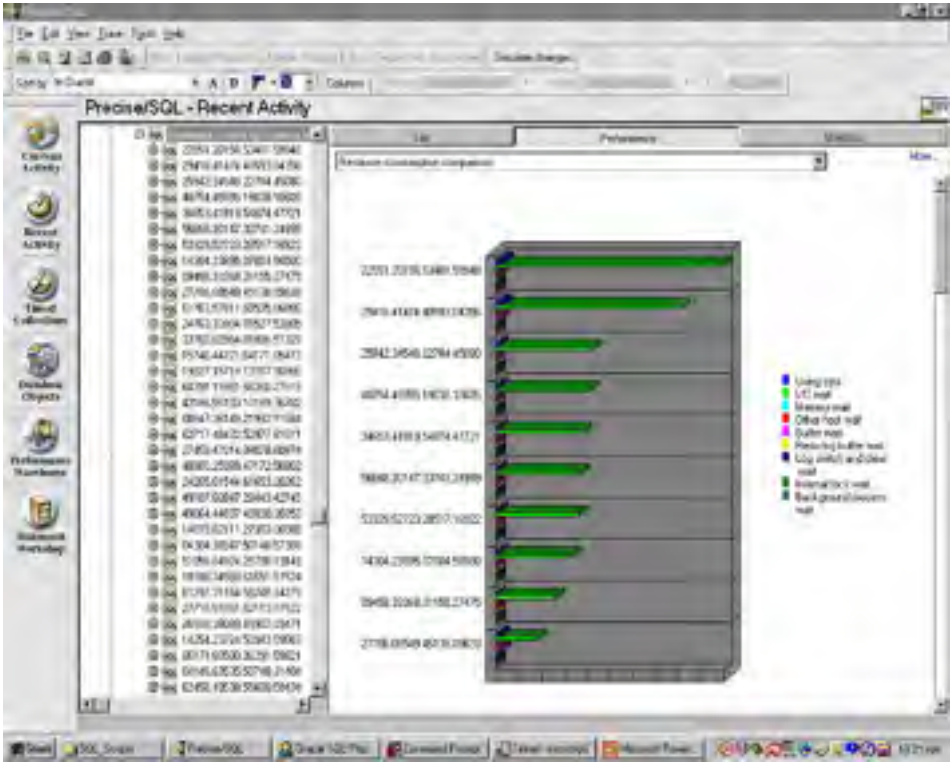


Figure 9: Example of Hotspotting

The thing to notice in Figure 9 is that every statement is showing IO wait as its major resource problem. In this particular the entire database was striped across eight disk drives and all files where placed into a single RAID 1 array. In addition many tablespace contained dozens if not hundreds of tables or indexes and in some cases partitions for a single object where all placed into the same tablespace. The system in figure 9 was also using an EMC drive array but due to the errors in file layout it could not take advantage of the EMC advantages.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Tablespace Segments & Free Space

If we had the mythical setup where a single table occupied a single tablespace we probably would not have a significant problem with fragmentation. Fragmentation occurs when multiple objects with differing segment extent sizes occupy the same tablespace and undergo dynamic extension and subsequent DBA or system actions result some of these objects being resized or removed. The action of resizing or removing segments from a tablespace results in empty spots, if there are multiple segments in the tablespace, these empty spots may be randomly distributed. If adjacent extents end up being freed, we can easily coalesce them into a single larger extent, but if there is intervening live extent areas, we cannot. These two fragmentation models are known as honeycomb, when the free extents are side by side, or swiss cheese, when the extents are separated by live segments.

Honeycomb fragmentation in a tablespace is automatically cleaned up by the smon process if the default storage parameter PCTINCREASE is set greater than zero. For tablespaces where PCTINCREASE is greater than zero use the ALTER TABLESPACE tablespace_name COALESCE command to coalesce adjacent areas of free space.

Some experts have proposed always sizing objects in a given tablespace using the same extent sizes or equal multiples of the same extent sizes. There is probably some merit to this since it will reduce the likelihood of a freed extent not being able to be reused, however, it will also result in some sparse objects that have wasted space.

I believe a blended approach is in order where the sizes for objects are calculated and then an attempt is made to choose and extent size for objects in the same tablespace that will reduce the overall number of extents per objects but also allow maximal reuse of space.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Let's examine some quick placement guidelines and move on:

- Put indexes and tables in as physically separate disk areas as possible
- Never put rollback segments in with data or index segments
- Place redo logs in their own area
- Separate highly active tables and indexes into their own tablespaces
- Group low activity tables and indexes together (tables with tables and indexes with indexes)
- Partition high activity tables and indexes to help balance disk IO and prevent hot disks.
- Use as many disk controller channels as is required to reduce channel saturation.

Tables & Indexes/Partitioning

With Oracle8 the capability to partition tables and indexes was introduced. Prior to Oracle8 partitioned views where possible but maintaining them was difficult. A partitioned table is partitioned by ranges of values. In Oracle8i the partitions can be subpartiioned based on a hash value as well. Indexes can be locally partitioned (partitioned identically to the tables) or can be globally partitioned (the index is partitioned on a different scheme than the table).

Partitioning is helpful where the table size is large and the values in the table can be readily separated into discrete ranges. An easy example is when sales are divided by ranges such as sales division, quarter or store. Partitioning does little good unless the partitions are placed on different access paths whether these are disks, arrays or symmetrix devices. If a table or index is properly partitioned then Oracle parallel operations can access the partitions

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

in an efficient manner. For some Oracle parallel operations the table must be partitioned.

Generally you want to examine partitioning candidates and determine what columns are most frequently used in queries involving ranges of values such as queries using BETWEEN, inequalities such as greater than and less than and using the IN clause to portion out a discrete data set. However, in some cases where a table is extremely large choosing a partition key that equally divides the table values can improve parallel scans by equalizing IO across query slave processes.

The V$ views

Throughout the next two days lectures you will hear about the V$ views. The V$ views provide a continually updated look at internal statistics. You may hear the V$ views called dynamic performance views or tables for this reason. There are approximately 143 or more V$ views in version 8.0.5. The views themselves are actually constructed on top of the C structs that are at the heart of the Oracle executable. The C structs in Oracle are named using a K$ or X$ prefix and general can't be seen by anyone but the SYS user and then only if a view is created against them. Figure 10 shows a sample of the layout of the V$ tables as extracted via a partial screen shot from the RevealNet Oracle Administrator Visual Dictionary

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

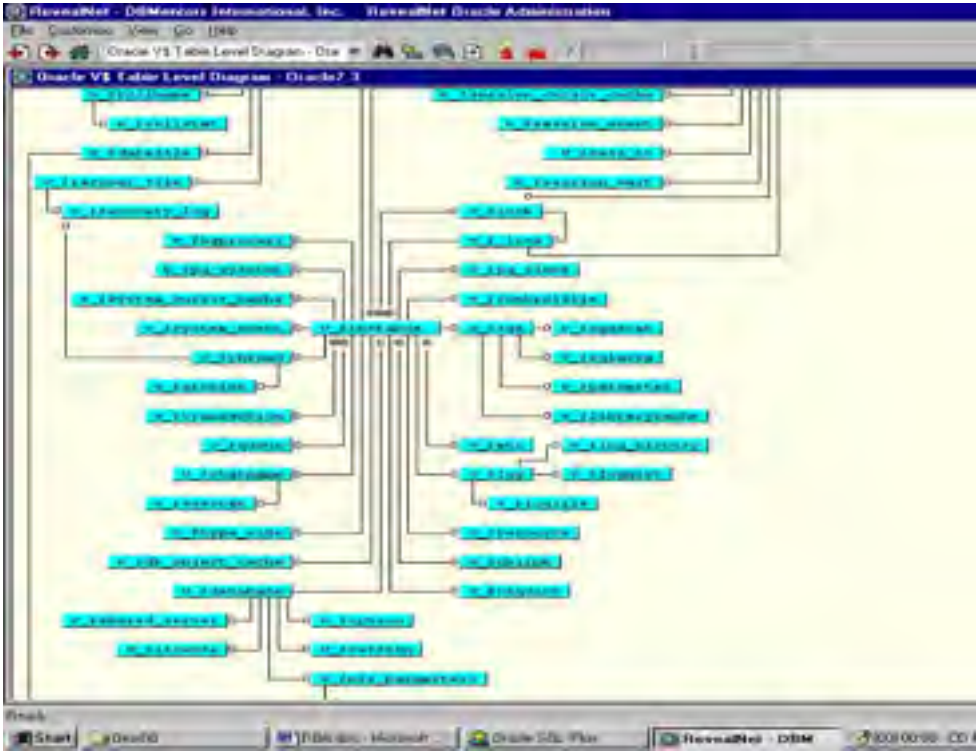


Figure 10: Example of the V$ view relations

How are they used?

The V$ views are used to examine current performance indicators. By using a delta method over several equally timed selects trending can be performed. This delta method is how many of the monitoring tools get their information about Oracle performance. As far as I know only one tool, Precise*SQL from Precise Software Solutions (also sold as DBTuner from EMC) samples externally directly from memory. Quest Software is working on externally sampling but I am not aware if they have it in production as of this writing.

The benefits of external sampling are that increased sample rates (up to 99 per second) can be obtained with little impact on the

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Oracle database. Internal sampling is prone to the same delays and problems any Oracle internal connection falls prey too, thus limiting the sustained sample ratre that can be obtained without seriously impacting performance.

A V$ view that is very useful for shared pool analysis is called the V$SQLAREA view and as its name implies it is used to show the contents of the shared SQL area in the shared pool. An example view based on the V$SQLAREA view and a companion report are show in Source 34 and 35.

```
CREATE OR REPLACE VIEW sql_garbage AS
SELECT
  b.username users,
  SUM(a.sharable_mem+a.persistent_mem) Garbage,
  TO_NUMBER(NULL) good
FROM
   sys.v_$sqlarea a,
   dba_users b
WHERE
  (a.parsing_user_id = b.user_id AND a.executions<=1)
GROUP BY b.username
UNION
SELECT DISTINCT
  b.username users,
  TO_NUMBER(NULL) garbage,
  SUM(c.sharable_mem+c.persistent_mem) Good
FROM
   dba_users b,
   sys.v_$sqlarea c
WHERE
  (b.user_id=c.parsing_user_id AND c.executions>1)
GROUP BY b.username;
```

Listing 34: Example view using the V$SQLAREA View

```
COLUMN garbage FORMAT A14 HEADING 'Non-Shared SQL'
COLUMN good FORMAT A14 HEADING 'shared SQL'
COLUMN good_percent FORMAT A14 HEADING 'Percent Shared'
COLUMN users FORMAT A14 HEADING Users
COLUMN nopr NOPRINT
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
SET FEEDBACK OFF
@title80 'Shared Pool Utilization'
SPOOL rep_out\&db\sql_garbage
SELECT 1 nopr,
a.users users,
TO_CHAR(a.garbage,'9,999,999,999') garbage,
TO_CHAR(b.good,'9,999,999,999') good,
TO_CHAR((b.good/(b.good+a.garbage))*100,'9,999,999.999')
good_percent
FROM sql_garbage a, sql_garbage b
WHERE a.users=b.users
AND a.garbage IS NOT NULL AND b.good IS NOT NULL
UNION
select 2 nopr,
'-------------' users,'--------------' garbage,'------------
--' good,
'--------------' good_percent FROM dual
UNION
SELECT 3 nopr,
TO_CHAR(COUNT(a.users)) users,
TO_CHAR(SUM(a.garbage),'9,999,999,999') garbage,
TO_CHAR(SUM(b.good),'9,999,999,999') good,
TO_CHAR(((SUM(b.good)/(SUM(b.good)+SUM(a.garbage)))*100),'9,
999,999.999') good_percent
FROM sql_garbage a, sql_garbage b
WHERE a.users=b.users
AND a.garbage IS NOT NULL AND b.good IS NOT NULL
ORDER BY 1,3 DESC
/
SPOOL OFF
CLEAR COLUMNS
```

Source 35: Example Report using View Based on V$SQLAREA

Example results from the report shown in Source 35 are shown in Listing 34.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
Date: 01/28/00
Page:   1
Time: 10:45 AM            Shared Pool Utilization
MAULT
                                dw database

Users           Non-Shared SQL Shared SQL     Percent Shared
--------------- -------------- -------------- --------------
ASMITH             371,387,006      1,007,366           .271
NETSPO              10,603,456        659,999          5.860
DCHUN                6,363,158        151,141          2.320
DSSUSER              5,363,057        824,865         13.330
MRCHDXD              4,305,330        600,824         12.246
DWPROC               2,690,086      4,901,400         64.564
CWOODY                 946,199        239,604         20.206
TTERIFIC               877,644         93,323          9.611
GCMATCH                604,369      1,637,788         73.045
MAULT                  445,566      3,737,984         89.350
PRECISE                205,564     46,342,150         99.558
BWITE                  154,754         35,858         18.812
SYS                    146,811      9,420,434         98.465
SSMITH                 102,460      8,523,746         98.812
MRCHPHP                 56,954         59,069         50.911
MRCHAEM                 42,465         65,017         60.491
--------------  -------------- -------------- --------------
16                 404,553,888     78,358,468         16.226
```

Listing 34: Example Output from the SQL GARBAGE Report

Other useful V$ views are shown in Table 11.

View Name	Description
V$SQLTEXT	Contains full SQL text of stored SQL area
V$SYSSTAT	Contains overall system level statistics
V$SESSTAT	Contains session statistics
V$SESSION	Contains information on current sessions
V$ROLLSTAT	Contains statistics on rollback segments

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

View Name	Description
V$ROLLNAME	Contains rollback segment names
V$SGA	Contains information on SGA areas (rollup from V$SGASTAT)
V$SGASTAT	Contains information on SGA segment sizes
V$INSTANCE	Contains information on the instance
V$PARAMETER	Contains settings for all initialization parameters
V$DB_OBJECT_CACHE	Contains information on the SGA Object Caches
V$LIBRARYCACHE	Contains information on the shared pool library caches
V$LOG	Contains information on all current redo logs
V$LOG_HIST(ORY)	Contains information on redo log switches
V$LOGFILE	Contains locations of all redo log files
V$FIXED_VIEW_DEFINITION	Contains the definitions for all V$ and GV$ views

Table 11: Views Useful for Monitoring and Tuning

Table 11 only lists those views I use regularly, you will in the presentation on tuning that several others are useful for specific monitoring of the database buffers and locks as well.

The Optimizers & the Analyze Command

Optimizers are required to ensure that a query is processed in an optimal manner. In earlier versions of Oracle the only available optimizer was the RULE based optimizer. In later versions of Oracle7 and all versions of Oracle8 both RULE and COST based optimizers are available.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

RULE Based Optimizer

As its name implies the rule based optimizer uses a set of 15 rules to determine how to best process a query. The rule chosen is based strictly on query structure and pays no attention to the number of rows, partitioning or other query specific features and statistics. The query optimization paths used by the cost based optimizer are:

Rank	Path
1	Single row by ROWID
2	Single row by Cluster join
3	Single row by hash cluster key with unique or primary key
4	Single row by unique or primary key
5	Cluster join
6	Hash cluster key
7	Indexed cluster key
8	Composite Index
9	Single Column index
10	Bounded range search on indexed columns
11	Unbounded range search on indexed columns
12	Sort-merge join
13	MAX or MIN on indexed column
14	ORDER BY indexed columns
15	Full table scan

The major benefit of the cost based optimizer was that for a specific query the execution plan was stable regardless of how big the table or its associated indexes grew as long as the index structure didn't change. The major disadvantages of the cost based optimizer are that the execution plan didn't change no matter how large the table or its indexes grew, and, it can't take advantage of new features as easily as a cost based optimizer.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

COST Based Optimizer

The COST based optimizer was introduced in later versions of Oracle7. The cost based optimizer uses a proprietary algorithm to assign a cost to each step of an execution plan for a query, and then, by looking at equivalent plans the optimizer can choose the one with the lowest cost for a specific query. The cost algorithm is dependent upon accurate table statistics in order to come up with accurate cost information. The table statistics are created and maintained by use of the ANALYZE command.

The major benefit of the cost based optimizer is that for changes in table and index sizes the optimizer can re-evaluate the execution plan and adjust to obtain an optimal performance profile. However, the cost based optimizer is only as good as the statistics fed to it, and, in tests a fully qualified DBA using the rule based optimizer can get better performance. However, unless you like to spend your free time using the tools tkprof, explain plan or third query party analyzers, I suggest just using the cost based optimizer and an "intelligent" table analyzer procedure such as the one shown in Source 36.

```
CREATE OR REPLACE PROCEDURE check_tables (
owner_name in varchar2,
break_rows in number,
percent_change in number) AS
--
CURSOR get_tab_count (own varchar2) IS
        SELECT table_name, nvl(num_rows,1)
        FROM dba_tables
        WHERE owner = upper(own);
--
  tab_name  VARCHAR2(64);
  rows            NUMBER;
  string          VARCHAR2(255);
  cur             INTEGER;
  ret             INTEGER;
  row_count       NUMBER;
  com_string      VARCHAR2(255);
--
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
BEGIN
  OPEN get_tab_count (owner_name);
  LOOP
   BEGIN
        FETCH get_tab_count INTO tab_name, rows;
        tab_name:=owner_name||'.'||tab_name;
        IF rows=0 THEN
        rows:=1;
        END IF;
   EXIT WHEN get_tab_count%NOTFOUND;
   DBMS_OUTPUT.PUT_LINE('Table name: '||tab_name||' rows:
'||to_char(rows));
   DBA_UTILITIES.GET_COUNT(tab_name,row_count);
   IF row_count=0 THEN
        row_count:=1;
   END IF;
   DBMS_OUTPUT.PUT_LINE('Row count for '||tab_name||':
'||to_char(row_count));
   DBMS_OUTPUT.PUT_LINE('Ratio:
'||to_char(row_count/rows));
        IF (row_count/rows)>(1+(percent_change/100))
         OR (rows/row_count)>(1+(percent_change/100)) THEN
         BEGIN
           IF (row_count<break_rows) THEN
              string :=
              'ANALYZE TABLE '||tab_name||' COMPUTE
STATISTICS ';
           ELSE
              string :=
       'ANALYZE TABLE '||tab_name||' ESTIMATE STATISTICS
SAMPLE 30 PERCENT';
           END IF;
           cur := DBMS_SQL.OPEN_CURSOR;
           DBMS_OUTPUT.PUT_LINE('Beginning analysis');
           DBMS_SQL.PARSE(cur,string,dbms_sql.v7);
           ret := DBMS_SQL.EXECUTE(cur)  ;
           DBMS_SQL.CLOSE_CURSOR(cur);
           DBMS_OUTPUT.PUT_LINE(' Table: '||tab_name||' had
to be analyzed.');
           EXCEPTION
            WHEN OTHERS THEN
             raise_application_error(-20002,
              'Error in analyze: '||to_char(sqlcode)||' on
'||tab_name,TRUE);
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
            IF dbms_sql.is_open(cur) THEN
             dbms_sql.close_cursor(cur);
            END IF;
          END;
         END IF;
     EXCEPTION
      WHEN others THEN
        null;
    END;
  COMMIT;
  END LOOP;
  CLOSE get_tab_count;
END;
```

Source 36: Example PL/SQL Procedure to do a “Smart” analysis of tables.

The procedure in Source 36 allows you to specify the schema that owns the tables to be analyzed, it then proceeds to check the actual row count and compare that value to the row count as stored in the DBA_TABLES view. The row count stored in the DBA_TABLES view is the calculated row count from the last analyze operation. If the percent difference (either positive or negative) exceeds the value input (I usually use 10) then the table will be analyzed. If the table exceeds a predetermined row threshold (depending on table row size and the size of your sort area) then an estimate statistics using 30% sampling is used, otherwise a full compute of statistics is performed.

Using a smart analysis procedure analysis time dropped from several hours to 30 to 40 minutes in one PeopleSoft environment. By only analyzing those tables that actually require it you can produce a considerable time saving. In the version of this program provided with the RevealNet Oracle Administrator program all activities are also logged in a database table for later review and use in trending analysis.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The Parallel Query Option

Introduced in later versions of Oracle7 the parallel query option (PQO) allows multiple processes to simultaneously fetch records and perform sorting operations. This parallelization of operations can lead to impressive speed improvements in a properly set up parallel environment.

The most important item to specify in a parallel environment is the number of parallel query slaves. Too few and you don't get the full benefits, too many and they end up competing with each other for resources. Of course parallel operations are of little or no benefit if your system doesn't have parallel processors and your tables aren't spread across multiple disks in a stripe set or partitioned. Along with the number of parallel query slaves the degree of parallel (DOP) for the tables and indexes needs to be set properly.

The maximum number of parallel query slaves should generally be set to at least twice the number of CPUs or to twice the number of disks that he object was spread across. The DOP can be determined by forcing a full table scan for tables (use a where 1=2 in a select count(*) from the table) and then timing the response for different DOP settings. These numbers are then adjusted up or down depending on performance.

Parallel query settings

There are several initialization parameters that pertain to parallel query:

parallel_adaptive_multi_user -- TRUE or FALSE, defaults to FALSE, when set to TRUE, enables an adaptive algorithm designed to improve performance in multi-user environments that use Parallel Query(PQ). It does this by automatically reducing the requested degree of parallelism based on the current number of active PQ users on the system. The effective degree of parallelism

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

will be based on the degree of parallelism from the table or hint divided by the total number of PQ users. The algorithm assumes that the degree of parallelism provided has been tuned for optimal performance in a single user environment.

This parameter will work best when used in single node Symmetric Multi-Processors(SMPs). However, it can be set to TRUE when using OPS if all of the following conditions are true:

- All Parallel Query users connect to the same node
- Instance Groups are not configured
- Each node has more than one CPU

On a system running OPS, if all of the above conditions are met, and the parameter is set to TRUE, if the system is not a Shared Nothing MPP, the algorithm will attempt to reduce the instances first, then the degree. If it is a Shared Nothing MPP, it will only reduce the degree. If any of the above conditions is not met, and the parameter is set to TRUE, the algorithm may reduce parallelism excessively, causing the system to become unnecessarily idle.

optimizer_percent_parallel -- specifies the amount of parallelism that the optimizer uses in its cost functions. The default of 0 means that the optimizer chooses the best serial plan. A value of 100 means that the optimizer uses each object's degree of parallelism in computing the cost of a full table scan operation. Low values favor indexes, and high values favor table scans.

Cost-based optimization will always be used for any query that references an object with a nonzero degree of parallelism. For such queries a RULE hint or optimizer mode or goal will be ignored. Use of a FIRST_ROWS hint or optimizer mode will override a nonzero setting of OPTIMIZER_PERCENT_PARALLEL.

parallel_min_percent -- specifies the minimum percent of threads required for parallel query. Setting this parameter ensures that a

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

parallel query will not be executed sequentially if adequate resources are not available. The default value of 0 means that this parameter is not used.

If too few query slaves are available, an error message is displayed and the query is not executed. Consider the following settings:

PARALLEL_MIN_PERCENT = 50
PARALLEL_MIN_SERVERS = 5
PARALLEL_MAX_SERVERS = 10

In a system with 20 instances up and running, the system would have a maximum of 200 query slaves available. If 190 slaves are already in use and a new user wants to run a query with 40 slaves (for example, degree 2 instances 20), an error message would be returned because 20 instances (that is, 50% of 40) are not available.

parallel_min_servers -- specifies the minimum number of query server processes for an instance. This is also the number of query server processes Oracle creates when the instance is started.

parallel_max_servers -- specifies the maximum number of parallel query servers or parallel recovery processes for an instance. Oracle will increase the number of query servers as demand requires from the number created at instance startup up to this value. The same value should be used for all instances in a parallel server environment.

Proper setting of the PARALLEL_MAX_SERVERS parameter ensures that the number of query servers in use will not cause a memory resource shortage during periods of peak database use.

If PARALLEL_MAX_SERVERS is set too low, some queries may not have a query server available to them during query processing.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Setting PARALLEL_MAX_SERVERS too high leads to memory resource shortages during peak periods, which can degrade performance. For each instance to which you do not want to apply the parallel query option, set this initialization parameter to zero.

If you have reached the limit of PARALLEL_MAX_SERVERS on an instance and you attempt to query a GV$ view, one additional parallel server process will be spawned for this purpose. This extra process will serve any subsequent GV$ queries until expiration of the PARALLEL_SERVER_IDLE_TIME, at which point the process will terminate. The extra process is not available for any parallel operation other than GV$ queries.

Note that if PARALLEL_MAX_SERVERS is set to zero for an instance, then no additional parallel server process will be allocated to accommodate a GV$ query.

parallel_server_idle_time -- specifies the amount of idle time after which Oracle terminates a process for parallel operations (parallel query, parallel DML, or parallel DDL). This value is expressed in minutes. The parameter must be set to 1 or greater for the query processes to terminate. 0 means the processes are never terminated.

parallel_execution_message_size -- specifies the size of messages for parallel execution (Parallel Query, PDML, Parallel Recovery, replication). The default value should be adequate for most applications. Typical values are 2148 or 4096 bytes. Larger values would require a larger shared pool.

parallel_min_message_pool – Defaults to (cpus*parallel_max_servers*1.5*(OS message buffer size) or cpus*5*1.5*(OS message size)). Specifies the minimum permanent amount of memory which will be allocated from the SHARED POOL, to be used for messages in parallel execution.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

This memory is allocated at startup time if PARALLEL_MIN_SERVERS is set to a non-zero value, or when the server is first allocated. Setting this parameter is most effective when PARALLEL_MIN_SERVERS is set to a non-zero value, because the memory will be allocated in a contiguous section.

This parameter should only be set if the default formula is known to be significantly inaccurate. setting this parameter too high will lead to a shortage of memory for the shared pool; setting it too low will lead to costlier memory allocation when doing parallel execution. This parameter cannot be set to a number higher than 90% of the shared pool.

parallel_transaction_resource_timeout -- is a Parallel Server parameter. This parameter specifies the maximum amount of time which can pass before a session, executing a parallel operation (either parallel DDL or parallel DML), times out while waiting for a resource held by another session in an incompatible lock mode. Such timeouts are an indication of potential deadlock involving the parallel transaction and other transactions currently running in the parallel
server system.

The value of the parameter is specified in seconds. The time begins when the session starts to wait for a busy resource. Setting the parameter to 0 sets the maximum timeout to an effectively infinite value. Note that this is only the maximum timeout allowed for parallel transactions. For certain resources, a lower timeout value is used by Oracle.

parallel_broadcast_enabled – Defaults to FALSE. Allows you to improve performance in certain cases involving hash and merge joins. When set to TRUE, if you are joining a very large join result set with a very small result set (size being measured in bytes, rather than number of rows), the optimizer has the option of broadcasting the row sources of the small result set, such that a

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

single table queue will send all of the small set's rows to each of the parallel servers which are processing the rows of the larger set. The result is enhanced performance.

shared_pool_size – The shared pool will have to be increased in size to accommodate the parallel query message areas and IO queues. I suggest that the LARGE POOL be designated to prevent PQO from causing shared pool problems.

large_pool_size – The large pool will automatically be configured at a minimum size of 600k if certain initialization parameters are set in Oracle8 (as discussed in previous lessons). I suggest manually setting the size.

Problems In Parallel Query Usage

- Setting too many parallel query slaves
- Setting too few parallel query slaves
- Not allocating enough shared pool or not properly configuring the large pool.
- Excess memory usage due to large numbers of parallel query slaves
- Multiple extent problems when creating objects (usually indexes) for objects with large DOP settings.

Security

Security in Oracle has until fairly recently been at the password level only. In recent releases proper security features such as password aging, resource allocation and row-level security have been added. Security begins with user management.

Users

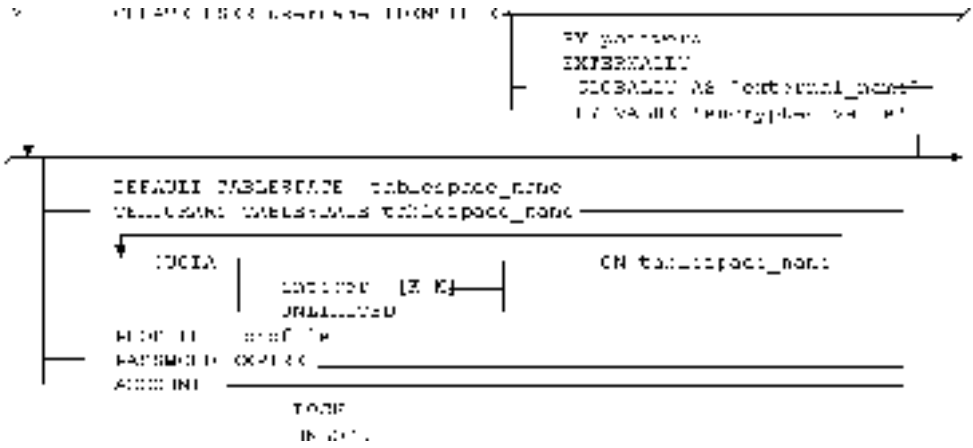
To access your database, an account must be created in the Oracle database for the user. The exceptions to this are the sys

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

and system users that are created by Oracle when the database is created. In the sections that follow, I will discuss the creation, alteration and dropping of users for the Oracle database.

Creating Users

To create a user, you must have the **CREATE USER** privilege. You can create users with Server Manager, Oracle Enterprise Manager or at the command line in SQL*Plus. The command syntax for creating a user is illustrated in Listing 35.



Listing 35: Syntax for creating a user.

Here is an example:

```
CREATE USER  james  IDENTIFIED BY   abc1
DEFAULT TABLESPACE  users
TEMPORARY TABLESPACE  temp
PASSWORD EXPIRE
QUOTA  1M ON  users
PROFILE  enduser  ;
```

You need to assign each new user a password or indicate that operating system authentication will be used. Passwords are stored

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

in the database in encrypted format and cannot be read by any user. The use of operating system authentication means that once your user has logged in at the operating system level, no username or password will be required when logging into the Oracle database. Users who are not assigned an Oracle password are designated as **IDENTIFIED EXTERNALLY**. Oracle depends on the operating system for authentication of the user. To use external authentication with other than the default prefix of **OPS$**, you must set the **OS_AUTHENT_PREFIX** in the database parameter file. In Oracle8 you can also create the user in a locked state or with a pre-expired password. The **BY VALUE** clause allows resetting of a users password to a previous encrypted value and is used when transferring a user manually from one database to another.

Password expiry options are set using a **PROFILE** which we will discuss later.

The **IDENTIFIED GLOBALLY** clause is used to tell Oracle that the user authentication is obtained from a central authority such as the Oracle Security Server.

When you create a user, you can designate a specific tablespace to be the default tablespace for that user. The designation of a default tablespace means that all the objects created by that user will be placed in that tablespace unless the user specifically indicates that the database object be placed in another tablespace. If no default tablespace is indicated for a user, the system tablespace will be the default for that user.

When you create a user, you can also designate a specific tablespace to be the temporary tablespace. This designation of a temporary tablespace specifies the tablespace that will be used for any database actions, which require use of a workspace for storage of intermediate results for actions such as sorting.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

If no temporary tablespace is indicated for a user, the system tablespace will be used. When you designate a default tablespace, temporary tablespace, or quota on a tablespace, this does not implicitly grant any system or object privileges on the default tablespace, the user is granted unlimited quota on the designated temporary tablespace. You can give a user permission to create objects in tablespaces with the **QUOTA** clause.

Note: I suggest that the only users that are left with the system tablespace as their default tablespace are users such as sys and dbsnmp that are created and maintained internally by Oracle processes. All users should have an explicitly assigned temporary tablespace to prevent sorting from occurring in the system tablespace.

As the DBA, you can access the **DBA_USERS** view for information on all users. Each user can access the **USER_USERS** view for information related to them. Table 12 shows the data stored in **DBA_USERS** and **USER_USERS**.

Column	Definition
DBA_USERS	
Username	Oracle login name for the user
user_id	Oracle unique user id number
Password	An encrypted password or IDENTIFIED EXTERNALLY
account_status	Tells account status when password expiry functions are used
lock_date	Date expiry function locked the account
expiry_date	Date password will expire if expiry parameters are set
default_tablespace	Tablespace assigned as the default for the user

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Column	Definition
Temporary_tablespace	Tablespace assigned for actions requiring a workspace
Created	Date the user was created within the Oracle database
USER_USERS	
Username	Oracle login name for the user
user_id	Oracle unique user id number
account_status	Tells account status when password expiry functions are used
lock_date	Date expiry function locked the account
expiry_date	Date password will expire if expiry parameters are set
default_tablespace	Tablespace assigned as the default for the user
Temporary_tablespace	Tablespace assigned for actions requiring a workspace
Created	Date the user was created within the Oracle database

Table 12: Data dictionary views for user data.

To enable a user to create objects in a tablespace, you need to specify a quota for that user on that tablespace. The tablespace quota may be limited to a specific amount of kilobytes or megabytes, or may be designated as unlimited. An unlimited quota indicates that the user can have any portion of a tablespace that is not already in use by another user. If the user is not assigned the **UNLIMITED TABLESPACE** system privilege and the assigned limit is reached, the user will no longer be able to create additional objects or insert rows into any objects he owns in that tablespace. One thing to remember is that the roles **RESOURCE** and **DBA** automatically grants **UNLIMITED TABLESPACE**, so only use them when absolutely required.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The **DBA_TS_QUOTAS** view provides tablespace quota information for all users in the database. The **USER_TS_QUOTAS** view provides tablespace quota information for the current user. When you query **DBA_TS_QUOTAS** or **USER_TS_QUOTAS**, a designation of -1 in the **max_bytes** and **max_blocks** columns indicates that the user has unlimited quota on that tablespace. Table 13 shows the data dictionary views associated with quotas.

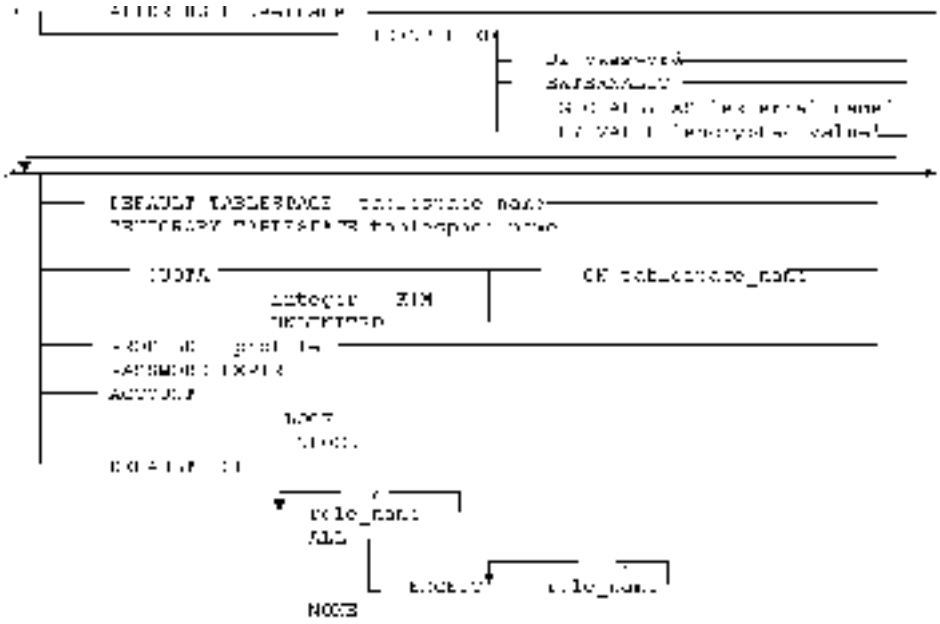
Column	Definition
DBA_TS_QUOTAS	
Tablespace_name	Name of the tablespace
Username	Name of the user
Bytes	Number of bytes assigned to that user
Blocks	Number of blocks assigned to that user
max_bytes	Maximum number of bytes allowed for the user or -1 for unlimited
max_blocks	Maximum number of blocks allowed for the user or -1 for unlimited
USER_TS_QUOTAS	
Tablespace_name	Name of the tablespace
Bytes	Number of bytes assigned to that user
Blocks	Number of blocks assigned to that user
max_bytes	Maximum number of bytes allowed for the user or -1 for unlimited
max_blocks	Maximum number of blocks allowed for the user or -1 for unlimited

Table 13: Quota data dictionary views.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Altering Users

To create a user, you must have the **ALTER USER** privilege. You can alter users with Server Manager or at the command line in SQL*Plus. The command line syntax for altering a user is shown in Figure 7.2.



Listing 35: Syntax for altering a user.

Here is an example:

```
ALTER  USER bill IDENTIFIED BY  xyz2
DEFAULT TABLESPACE  users
TEMPORARY TABLESPACE temp
QUOTA  1M  ON  users
PROFILE  enduser
DEFAULT ROLE ALL
ACCOUNT UNLOCK;
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

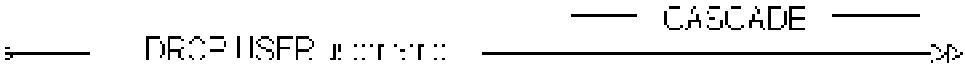
After a user is created, the only thing that you cannot alter for that user is the username. The password, default tablespace, temporary tablespace, the quota on a tablespace, profile, locked state, global name, password expire state and default role can all be altered by someone with the **ALTER USER** system privilege.

Each user can alter the Oracle password you initially assigned to that user when you created him, provided the user is not identified externally (via the operating system). In addition to the end user, users with the **ALTER USER** system privilege can issue the **ALTER USER** command to change the user's password. The use of operating system authentication can also be changed by a user with the **ALTER USER** system privilege. Any changes to the password will take effect the next time that user logs into Oracle.

When you change the default tablespace for a user, all future objects created by that user will be created in the new default tablespace you designated (unless otherwise specified by the user at the time the object is created). Remember the user must have a quota in the tablespace to create new objects in that tablespace. If a user reaches the maximum number of bytes assigned (specified in the quota), only a user with the **ALTER USER** system privileges will be able to increase the quota limit on the user.

Dropping Users

To drop a user, you must have the **DROP USER** system privilege. You can drop users with Server Manager or at the command line in SQL*Plus. The command line syntax for dropping a user is illustrated in Listing 36.



Listing 36: DROP USER Command

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Here is an example:

```
DROP USER edward CASCADE;
```

If a user owns any database objects, you can only drop that user by including the **CASCADE** keyword in the **DROP USER** command. The **DROP USER** command with the **CASCADE** keyword drops the user and all objects owned by that user. If you are using Server Manager to drop a user, you need to indicate that the associated schema objects be included in the command to drop the user. If a user owns objects and you fail to include **CASCADE**, you will receive an error message and the user will not be dropped. If a user is currently connected to the database, you cannot drop that user until he exits. After a user is dropped, all information on that user and all objects owned by that user are removed from the database.

After you have issued the command to drop a user, you cannot perform a rollback to re-create the user and his objects. **DROP USER** is a DDL command, which cannot be rolled back.

If you need the objects created by that user, you can revoke the **CREATE SESSION** system privilege to prevent the user from logging on, instead of dropping the user. You can also copy the objects to another user by importing the objects from an export made before the user was dropped. To avoid the problem of dropping a user without losing your application tables, all application tables should be owned by a separate application schema instead of an actual database user schema.

Grants

There are two types of (grants) privileges that can be granted:

- System privileges
- Object privileges

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

System privileges enable a user to perform a particular systemwide action or to perform a particular action on a particular type of object. For example, the privilege to create a table (**CREATE TABLE**) or insert rows into any table (**INSERT ANY TABLE**) are system privileges.

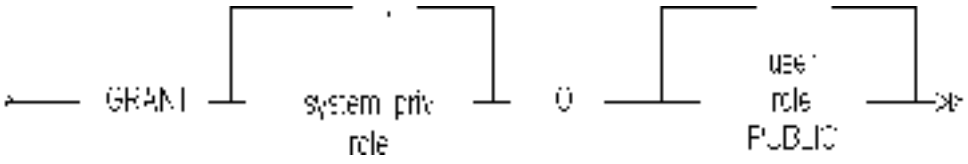
Object privileges enable a user to perform a particular action on a specific object, including tables, views, sequences, procedures, functions, and packages. For example, the privilege to insert rows into a particular table is an object privilege. Object privilege grants always include the name of the object for which the privilege is granted. Object privileges extend down to the level of individual columns in a table. These privileges are discussed in the following sections.

System Privileges

All users require the **CREATE SESSION** privilege to access the database. This privilege is automatically granted to all users when you perform the grants using Server Manager. If you create the user in command line mode, you must remember to explicitly grant each user the **CREATE SESSION** system privilege either directly or through a role. Listing 36 shows the syntax for the **GRANT** command. Here is an example of the command used to grant **CREATE SESSION** and **CREATE TABLE**.

```
GRANT  create session, create table
TO  annie
WITH ADMIN OPTION;
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.



Listing 36: The syntax for **GRANT** command

System privileges also include roles that have been created.

System privileges can be granted to other users when the grant made includes the **WITH ADMIN OPTION**.

There are over 80 distinct privileges. Most of these are self-explanatory. Table 13 lists of all the system privileges, as listed in RevealNet's Oracle Administration product (used with their permission).

Privilege	**Description**
System Privilege	Allows grantee to . . .
FOR CLUSTERS:	
CREATE CLUSTER	Create clusters in grantee's schema.
CREATE ANY CLUSTER	Create a cluster in any schema except SYS. Behaves similarly to CREATE ANY TABLE.
ALTER ANY CLUSTER	Alter clusters in any schema except SYS.
DROP ANY CLUSTER	Drop clusters in any schema except SYS.
FOR CONTEXTS:	
CREATE ANY CONTEXT	Create any context namespace.
DROP ANY CONTEXT	Drop any context namespace.
FOR DATABASE:	

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Privilege	Description
System Privilege	Allows grantee to . . .
ALTER DATABASE	Alter the database.
ALTER SYSTEM	Issue ALTER SYSTEM statements.
AUDIT SYSTEM	Issue AUDIT sql_statements statements.
FOR DATABASE LINKS:	
CREATE DATABASE LINK	Create private database links in grantee's schema.
CREATE PUBLIC DATABASE LINK	Create public database links.
DROP PUBLIC DATABASE LINK	Drop public database links.
FOR DIRECTORIES :	
CREATE ANY DIRECTORY	Create directory database objects.
DROP ANY DIRECTORY	Drop directory database objects.
FOR INDEXES:	
CREATE INDEX	Create in the grantee's schema an index on any table in the grantee's schema or a domain index.
CREATE ANY INDEX	Create in any schema except SYS a domain index or an index on any table in any schema except SYS.
ALTER ANY INDEX	Alter indexes in any schema except SYS.
DROP ANY INDEX	Drop indexes in any schema except SYS.
FOR LIBRARIES:	
CREATE LIBRARY	Create external procedure/function libraries in grantee's schema.
CREATE ANY LIBRARY	Create external procedure/function libraries in any schema except SYS.
DROP LIBRARY	Drop external procedure/function libraries in the grantee's schema.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Privilege	Description
System Privilege	Allows grantee to . . .
DROP ANY LIBRARY	Drop external procedure/function libraries in any schema except SYS.
FOR PROCEDURES :	
CREATE PROCEDURE	Create stored procedures, functions, and packages in grantee's schema
CREATE ANY PROCEDURE	Create stored procedures, functions, and packages in any schema except SYS.
ALTER ANY PROCEDURE	Alter stored procedures, functions, or packages in any schema except SYS
DROP ANY PROCEDURE	Drop stored procedures, functions, or packages in any schema except SYS
EXECUTE ANY PROCEDURE	Execute procedures or functions (standalone or packaged) Reference public package variables in any schema except SYS
FOR PROFILES:	
CREATE PROFILE	Create profiles
ALTER PROFILE	Alter profiles
DROP PROFILE	Drop profiles
FOR ROLES:	
CREATE ROLE	Create roles
ALTER ANY ROLE	Alter any role in the database
DROP ANY ROLE	Drop roles
GRANT ANY ROLE	Grant any role in the database
FOR ROLLBACK SEGMENTS:	
CREATE ROLLBACK SEGMENT	Create rollback segments
ALTER ROLLBACK SEGMENT	Alter rollback segments
DROP ROLLBACK SEGMENT	Drop rollback segments

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Privilege	Description
System Privilege	Allows grantee to . . .
FOR SEQUENCES:	
CREATE SEQUENCE	Create sequences in grantee's schema
CREATE ANY SEQUENCE	Create sequences in any schema except SYS
ALTER ANY SEQUENCE	Alter any sequence in the database
DROP ANY SEQUENCE	Drop sequences in any schema except SYS
SELECT ANY SEQUENCE	Reference sequences in any schema except SYS
FOR SESSIONS:	
CREATE SESSION	Connect to the database
ALTER RESOURCE COST	Set costs for session resources
ALTER SESSION	Issue ALTER SESSION statements
RESTRICTED SESSION	Logon after the instance is started using the SQL*Plus STARTUP RESTRICT statement
FOR SNAPSHOTS:	
CREATE SNAPSHOT	Create snapshots in grantee's schema
CREATE ANY SNAPSHOT	Create snapshots in any schema except SYS
ALTER ANY SNAPSHOT	Alter any snapshot in the database
DROP ANY SNAPSHOT	Drop snapshots in any schema except SYS
FOR SYNONYMS:	
CREATE SYNONYM	Create synonyms in grantee's schema
CREATE ANY SYNONYM	Create private synonyms in any schema except SYS
CREATE PUBLIC SYNONYM	Create public synonyms

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Privilege	Description
System Privilege	Allows grantee to . . .
DROP ANY SYNONYM	Drop private synonyms in any schema except SYS
DROP PUBLIC SYNONYM	Drop public synonyms
FOR TABLES:	
CREATE ANY TABLE	Create tables in any schema except SYS. The owner of the schema containing the table must have space quota on the tablespace to contain the table.
ALTER ANY TABLE	Alter any table or view in the schema
BACKUP ANY TABLE	Use the Export utility to incrementally export objects from the schema of other users
DELETE ANY TABLE	Delete rows from tables, table partitions, or views in any schema except SYS
DROP ANY TABLE	Drop or truncate tables or table partitions in any schema except SYS
INSERT ANY TABLE	Insert rows into tables and views in any schema except SYS
LOCK ANY TABLE	Lock tables and views in any schema except SYS
UPDATE ANY TABLE	Update rows in tables and views in any schema except SYS
SELECT ANY TABLE	Query tables, views, or snapshots in any schema except SYS
FOR TABLESPACES:	
CREATE TABLESPACE	Create tablespaces
ALTER TABLESPACE	Alter tablespaces
DROP TABLESPACE	Drop tablespaces
MANAGE TABLESPACE	Take tablespaces offline and online and begin and end tablespace backups

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Privilege	Description
System Privilege	Allows grantee to . . .
UNLIMITED TABLESPACE	Use an unlimited amount of any tablespace. This privilege overrides any specific quotas assigned. If you revoke this privilege from a user, the user's schema objects remain but further tablespace allocation is denied unless authorized by specific tablespace quotas. You cannot grant this system privilege to roles.
FOR TRIGGERS:	
CREATE TRIGGER	Create a database trigger in grantee's schema
CREATE ANY TRIGGER	Create database triggers in any schema except SYS
ALTER ANY TRIGGER	Enable, disable, or compile database triggers in any schema except SYS
DROP ANY TRIGGER	Drop database triggers in any schema except SYS
ADMINISTER DATABASE TRIGGER	Create a trigger on DATABASE. (You must also have the CREATE TRIGGER or CREATE ANY TRIGGER privilege.)
FOR TYPES:	
CREATE TYPE	Create object types and object type bodies in grantee's schema
CREATE ANY TYPE	Create object types and object type bodies in any schema except SYS
ALTER ANY TYPE	Alter object types in any schema except SYS
DROP ANY TYPE	Drop object types and object type bodies in any schema except SYS

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Privilege	Description
System Privilege	Allows grantee to . . .
EXECUTE ANY TYPE	Use and reference object types and collection types in any schema except SYS, and invoke methods of an object type in any schema if you make the grant to a specific user. If you grant EXECUTE ANY TYPE to a role, users holding the enabled role will not be able to invoke methods of an object type in any schema.
FOR USERS:	
CREATE USER	Create users. This privilege also allows the creator to assign quotas on any tablespace, set default and temporary tablespaces, assign a profile as part of a CREATE USER statement
ALTER USER	Alter any user. This privilege authorizes the grantee to Change another user's password or authentication method, Assign quotas on any tablespace, Set default and temporary tablespaces, and Assign a profile and default roles
BECOME USER	Become another user. (Required by any user performing a full database import.)
DROP USER	Drop users
FOR VIEWS:	
CREATE VIEW	Create views in grantee's schema
CREATE ANY VIEW	Create views in any schema except SYS
DROP ANY VIEW	Drop views in any schema except SYS

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Privilege	Description
System Privilege	Allows grantee to . . .
MISCELLANEOUS PRIVILEGES:	
ANALYZE ANY	Analyze any table, cluster, or index in any schema except SYS
AUDIT ANY	Audit any object in any schema except SYS using AUDIT schema_objects statements
COMMENT ANY TABLE	Comment on any table, view, or column in any schema except SYS
FORCE ANY TRANSACTION	Force the commit or rollback of any in-doubt distributedtransaction in the local database Induce the failure of adistributed transaction
FORCE TRANSACTION	Force the commit or rollback of grantee's in-doubt distributed transactions in the local database
GRANT ANY PRIVILEGE	Grant any system privilege.
SYSDBA	Perform STARTUP and SHUTDOWN operations, ALTER DATABASE: open, mount, back up, or change character set, CREATE DATABASE ARCHIVELOG and RECOVERY, Includes the RESTRICTED SESSION privilege
SYSOPER	Perform STARTUP and SHUTDOWN operations, ALTER DATABASE OPEN/MOUNT/BACKUP, ARCHIVELOG and RECOVERY Includes the RESTRICTED SESSION privilege

Table 13: Oracle database system privileges

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

As the DBA, you can access the **DBA_SYS_PRIVS** view for information on the system privileges granted to users. The format of this view is shown in Table 14.

Column	Definition
grantee	Oracle login name or role that received the privilege
privilege	The system privilege granted to the user or role
admin_option	Indicates YES if the grantee can pass along the privilege and NO if the grantee cannot pass along the system privilege

Table 14: Contents of the **DBA_SYS_PRIVS** data dictionary view.

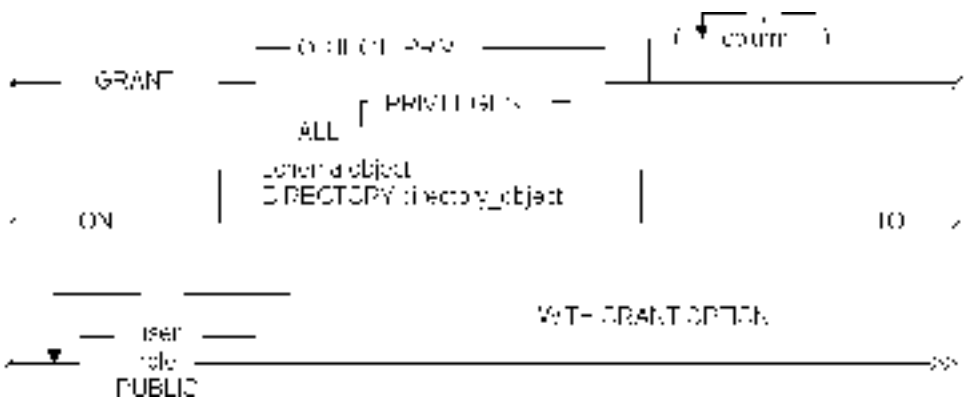
Users can see information related to them by accessing the corresponding user view: **USER_SYS_PRIVS**.

Object Privileges

Object privileges define a user's rights on existing database objects. All grants on objects take effect immediately.

To grant an object privilege, you must be the owner of the object, have been granted **WITH GRANT OPTION** on that object for that privilege, or have the system privilege **GRANT ANY PRIVILEGE**. You can also grant access to all users by granting the privilege to **PUBLIC**. Listing 37 shows the syntax for the **GRANT** command used to grant a table-level object privilege.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.



Listing 37: Syntax for the **GRANT** command used for a table-level or snapshot-level grant.

Here is an example:

```
GRANT select
ON  bob.emp
TO  derek;
```

As the DBA, you can access the **DBA_TAB_PRIVS** view for information on the object privileges granted to users. You should note that although it is named **DBA_TAB_PRIVS**, it also includes information on views and sequences, as well as tables. Table 14 shows the contents of this view.

Column	Definition
grantee	Oracle login name or role that received the privilege
owner	Owner of the table
table_name	Name of the table, view, or sequence
grantor	Oracle login name of the person granting the privilege
privilege	System privilege granted to the user

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Column	Definition
grantable	Indicates YES if the grantee can pass along the privilege and NO if the grantee cannot pass along the system privilege

Table 14: Contents of the **DBA_TAB_PRIVS** data dictionary view.

Users can see information on objects where they are the owner, grantor, or grantee by accessing the corresponding user view **USER_TAB_PRIVS**. Users can see information for all objects where that user or **PUBLIC** is the grantee with the **ALL_TAB_PRIVS** view. The **ALL_TAB_PRIVS** view is slightly different than the **USER_TAB_PRIVS** or **DBA_TAB_PRIVS** view. The contents of the **ALL_TAB_PRIVS** view are shown in Table 15.

Column	Definition
grantee	Oracle login name or role that received the privilege
grantor	Oracle login name of the person granting the privilege
owner	Owner of the table
table_schema	Schema of the object owner
table_name	Name of the table, view, or sequence
privilege	System privilege granted to the user
grantable	Indicates YES if the grantee can pass along the privilege and NO if the grantee cannot pass along the system privilege

Table 15: Contents of the **ALL_TAB_PRIVS** data dictionary view.

A table owner can grant the following object privileges to other users:

- ALTER
- DELETE

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- INDEX
- INSERT
- REFERENCES
- SELECT
- UPDATE
- EXECUTE (for stored functions, procedures and packages)

All grants on objects and revoking of those grants are valid immediately, even if a user is currently logged into the database. The **SELECT** privilege can only be granted on tables, views, and snapshots. The **EXECUTE** privilege is used for packages, procedures and functions. Remember that packages, procedures and functions are always executed with the permissions of the owner of that package, procedure or function.

By granting other users **INSERT**, **UPDATE**, **DELETE**, and **SELECT** privileges on your table, you allow them to perform that action on the table. By granting the **ALTER** privilege, you can allow another user to modify the structure of your table or create a trigger on your table. By granting users the **INDEX** privilege, you can allow them to create indexes on your table.

The **REFERENCES** privilege differs from the other privileges in that is does not actually grant the capability to change the table or data contained in the table. The **REFERENCES** privilege allows users to create foreign key constraints that reference your table.

Users can access the **USER_TAB_PRIVS_RECD** for information on table privileges where that user is the grantee. The corresponding **ALL_TAB_PRIVS_RECD** view includes all grants on objects where that user or **PUBLIC** is the grantee. Table 16 shows the contents of the **USER_TAB_PRIVS_RECD** view.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Column	Definition
owner	Owner of the table
table_name	Name of the table, view, or sequence
grantor	Oracle login name of the person granting the privilege
privilege	System privilege granted to the user
grantable	Indicates YES if the grantee can pass along the privilege and NO if the grantee cannot pass along the object privilege

Table 16: Contents of the **USER_TAB_PRIVS_RECD** data dictionary view.

Users can access the **USER_TAB_PRIVS_MADE** for information on table privileges that they have granted to others. The corresponding **ALL_TAB_PRIVS_MADE** view includes information on all the grants that user has made, as well as grants by others on that user's objects. Table 17 shows the contents of the **USER_TAB_PRIVS_MADE** view.

Column	Definition
grantee	Oracle user granted the privilege
table_name	Name of the table, view, or sequence
grantor	Oracle login name of the person granting the privilege
privilege	System privilege granted to the user
grantable	Indicates YES if the grantee can pass along the privilege and NO if the grantee cannot pass along the object privilege

Table 17: Contents of the **USER_TAB_PRIVS_MADE** data dictionary view.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Column Privileges

Only **INSERT**, **UPDATE**, and **REFERENCES** privileges can be granted at the column level. When granting **INSERT** at the column level, you must include all the not null columns in the row.

Here is an example:

```
GRANT  update (emp_name)
ON  edwin.emp
TO  joan;
```

As the DBA, you can access the **DBA_COL_PRIVS** view for information on the column-level object privileges granted to users. Table 18 shows the contents of the **DBA_COL_PRIVS** view.

Column	Definition
grantee	Oracle login name or role which received the privilege
owner	Owner of the table
table_name	Name of the table
column_name	Name of the column
grantor	Oracle login name of the person granting the privilege
privilege	System privilege granted to the user
grantable	Indicates YES if the grantee can pass along the privilege and NO if the grantee cannot pass along the object privilege

Table 18: Contents of the **DBA_COL_PRIVS** data dictionary view.

Users can access the **USER_COL_PRIVS_RECD** for information on column-level object privileges that have been granted to them. The **ALL_COL_PRIVS_RECD** includes information on all column privileges that have been granted to them or to **PUBLIC**. The format of the **USER_TAB_PRIVS_RECD** view is shown in Table 19.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Column	Definition
owner	Owner of the table
table_name	Name of the table, view, or sequence
column_name	Name of the column
grantor	Oracle login name of the person granting the privilege
privilege	System privilege granted to the user
grantable	Indicates YES if the grantee can pass along the privilege and NO if the grantee cannot pass along the column-level object privilege

Table 19: **USER TAB_PRIVS_RECD** data dictionary view.

Users can access the **USER_COL_PRIVS_MADE** for information on column privileges that they have granted to others. The corresponding **ALL_COL_PRIVS_MADE** includes information on all columns where the user is the owner or the grantor. The contents of the **USER_COL_PRIVS_MADE** view are shown in Table 20.

Column	Definition
grantee	Oracle user granted the privilege
table_name	Name of the table
column_name	Name of the column
grantor	Oracle login name of the person granting the privilege
privilege	System privilege granted to the user
grantable	Indicates YES if the grantee can pass along the privilege and NO if the grantee cannot pass along the column-level object privilege

Table 20: **USER_COL_PRIVS_MADE** data dictionary view.

Users can access information on all columns where they are the grantor, grantee, or owner, or where access has been granted to

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

PUBLIC with the corresponding **ALL_TAB_PRIVS_MADE** and **ALL_TAB_PRIVS_RECD** views.

View Grants

Views can have the **SELECT, INSERT, UPDATE** and **DELETE** grants issued against them. In order to perform **SELECT, INSERT, UPDATE** or **DELETE** operations against views (where it is allowed) you must grant the privileges for the underlying tables to the users you wish to have these privileges.

The information on grants made to views is located in the same views as for tables.

Other Grants

The only allowed grant for sequences is **SELECT**. For procedures, functions, packages, libraries and user defined types you may only grant **EXECUTE** privileges. The only allowed grant for a directory is **READ**, it is the only object which has a **READ** grant.

Revoking Grants

When system privileges are passed to others using the **WITH ADMIN OPTION**, revoking the system privileges from the original user will not cascade. The system privileges granted to others must be revoked directly. In contrast, when object privileges are passed on to others using the **WITH GRANT OPTION**, the object privileges are revoked when the grantor's privileges are revoked.

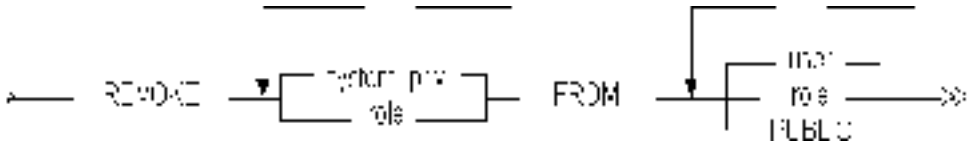
It is important to note that only object privileges will cascade when revoked; system privileges will not.

When the **WITH ADMIN OPTION** or **WITH GRANT OPTION** has been included in a grant to another user, the privilege cannot be revoked directly. You must revoke the privilege and then issue

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

another grant without the **WITH ADMIN OPTION** or **WITH GRANT OPTION**.

The command line syntax for revoking a system privilege is seen in Listing 37



Listing 37: Syntax for revoking a system privilege.

In this format roles are counted the same as system privileges and are also grouped with users.

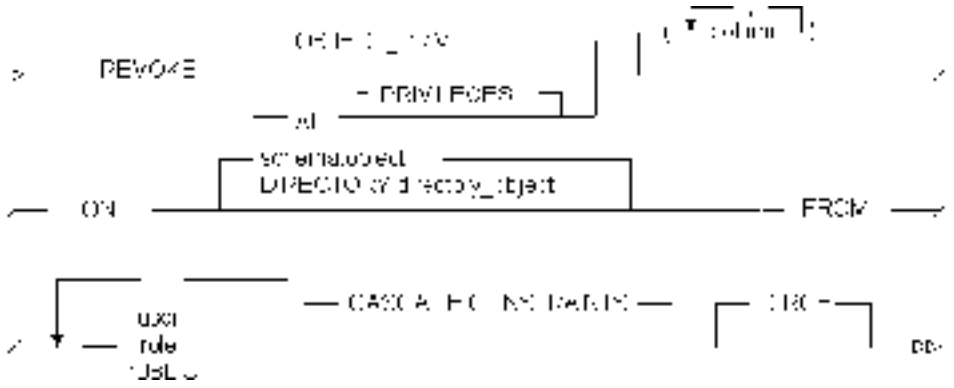
Here are some examples:

```
REVOKE   create table
FROM   judy;

REVOKE create table
FROM developer_role;

REVOKE dba
FROM monitor_role;
```

To revoke an object privilege, you must either be the owner of the object, have granted that privilege to that user with the **WITH GRANT OPTION**, or have the **GRANT ANY PRIVILEGE** system privilege**.**

You can revoke object and system privileges with Server Manager or at the command line in SQL*Plus. The command line syntax for revoking an object privilege is seen in Listing 38.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.



Listing 38 Syntax for revoking an object privilege.

Here is an example:

```
REVOKE  select
ON  mike.emp
FROM  stan;
```

When the object privilege **REFERENCES** has been granted, you must specify **CASCADE CONSTRAINTS** to drop the foreign key constraints that where created.

Use Of Roles

Using roles has several benefits, including:

- Reducing the number of grants and thereby making it easier to manage security.
- Dynamically changing the privileges for many users with a single grant or revoke.
- Selectively enabling or disabling depending on the application.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Roles can be used for most system and object privileges. Privileges granted through a role cannot be used for creating an object (views, packages, procedures, and functions). You need to grant privileges directly to the user for this.

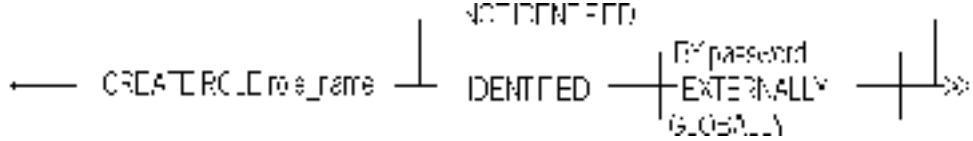
Creating Roles

You need to create the role first and then grant system and object privileges to that role. When you create the role, there are three password options available:

- No authentication
- Operating system authentication
- Password authentication

You can set operating system authentication either when the role is created or by using the database initialization parameters **OS_ROLES=TRUE** and **REMOTE_OS_ROLES=TRUE**. If you are using the multithreaded server option, you cannot use operating system authentication for roles.

To create a role, you must have the **CREATE ROLE** system privilege. You can create roles with Server Manager or at the command line in SQL*Plus. The command syntax for creating a role is seen in Listing 39.



Listing 39: Syntax for creating a role.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

A role can also be identified globally by using the **GLOBALLY** keyword in the **IDENTIFIED** clause. This means the role will be authenticated by the Oracle Security Server.

Here is an example:

```
CREATE  ROLE   appusers
GLOBALLY  IDENTIFIED;
```

To alter a role, you must have the **ALTER ANY ROLE** system privilege or have been granted the role with the **WITH ADMIN OPTION**. The creator of any role automatically has the **WITH ADMIN OPTION** for that role.

Grants To Roles

To grant a role to a user, you must either be the creator of that role or have the **GRANT ANY ROLE** privilege. You can grant roles to users with Server Manager or at the command line in SQL*Plus. Grants to roles will not take effect for a user if that user is currently logged into the database with that role. When the user exits or sets another role, the changes will take effect. Once roles have been granted to a user, they can be enabled and disabled.

Here is an example:

```
GRANT  enduser
TO  patrick ;
```

Here is an example for a role:

```
GRANT  create  session
TO   enduser;
```

Here is an example of granting an object privilege to a role:

```
GRANT  select
ON  john.emp
TO  enduser;
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The only system privilege which cannot be granted to a role is the **UNLIMITED TABLESPACE** grant, however, it is implicitly granted whenever the **DBA** or **RESOURCE** role grant is made to a user. Grants on objects can be passed to other users or to roles if the grantee has been given the **WITH GRANT OPTION**. However, you cannot assign a privilege that includes the **WITH GRANT OPTION** to a role. The **INDEX** and **REFERENCES** privileges cannot be granted to a role; they must be granted only to a user. You can grant that role to a user or to another role. However, you cannot grant a role to itself.

You can look at the data dictionary tables shown in Table 21 for information on views for roles.

Column	Definition
DBA_ROLES	
Role	Name of the role
password_required	Yes, No, Global or External for operating system authentication
DBA_ROLE_PRIVS	
grantee	Name of the user or role receiving the grant
granted_role	Name of the role
admin_option	Y if it was granted with the admin option
default_role	Y if this is the grantee's default role
ROLE_ROLE_PRIVS	
role	Name of the role receiving the role grant
granted_role	Name of the role granted the role
admin_option	Indicates the role was granted with the admin option

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Column	Definition
ROLE_SYS_PRIVS	
role	Name of the role receiving the system privilege
privilege	System privilege being granted
admin_option	Indicates the grant was with the admin option

Table 21: Data dictionary views for roles.

The **ROLE_TAB_PRIVS** view shown in Table 22 provides information on tables and column grants to roles.

Columns	Definition
role	Name of the role
owner	Owner of the table, view, or sequence
table_name	Name of the table, view, or sequence
column_name	Name of the column within the table, if applicable
privilege	Object privilege granted to that role
grantable	YES if it was granted with the admin option; NO otherwise

Table 22: Contents of the **ROLE_TAB_PRIVS** data dictionary view.

Setting Roles

When a user is created, the default for active roles is set to **ALL**. The default **ALL** means that all the roles granted to a user are active. The DBA can change the default with an **ALTER USER** command. A user can enable multiple roles at one time and use the **SET ROLE** command to switch between roles or activate all roles with the command **SET ROLE ALL**. The **SET ROLE ALL** command will not work if any of the roles assigned to that user requires either a password or operating system authentication. The command line syntax for setting roles is shown in Figure 13.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

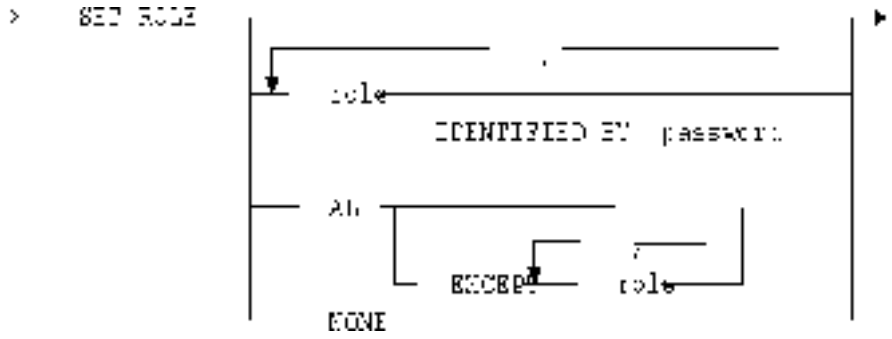


Figure 7.13 Syntax for setting roles

Users can look at the **SESSION_ROLES** view to find the roles that are currently enabled for them. Users can look at **SESSION_PRIVS** view to see the privileges available to their session.

If you determine that all control of roles will be at the operating system level, you can set the database initialization parameter **OS_ROLES** equal to **TRUE**. All roles must still be created first in the database. Any grants you previously made using the database command line or Server Manager are still listed in the data dictionary, but they cannot be used and are not in effect. If the use of roles is determined at the operating system level, the multithreaded server option cannot be used.

You can use the **MAX_ENABLED_ROLES** parameter in the database initialization file to set the number of roles that you will allow any user to have enabled at one time.

Special Roles

If you intend for administration level users to use OS authentication create the following two roles when you install the Oracle executables:

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- OSOPER
- OSDBA

Oracle creates the following three roles when you create the database:

- CONNECT
- RESOURCE
- DBA

When you execute the sql.bsq script, the following two roles are created:

- EXP_FULL_DATABASE
- IMP_FULL_DATABASE

When recovery manager is created, the following five roles are created:

- SELECT_CATALOG_ROLE
- EXECUTE_CATALOG_ROLE
- DELETE_CATALOG_ROLE
- RECOVER_CATALOG_ROLE
- RECOVERY_CATALOG_OWNER

Finally, with the addition of the advanced queuing option, the following two roles are created

- AQ_ADMINISTRATOR_ROLE
- AQ_USER_ROLE

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

In the following section, I'll explain these special roles and how they are used for database maintenance.

OSOPER And OSDBA

The OSOPER and OSDBA roles are created at the operating system level when Oracle is installed. They cannot be granted. The OSOPER and OSDBA roles are needed to perform database operations when the database is not mounted and therefore the data dictionary is not accessible. It is the OSOPER and OSDBA roles that are used when you use **CONNECT INTERNAL** to connect to the database using Server Manager.

The OSOPER role can perform the following database management commands:

- STARTUP
- SHUTDOWN
- ALTER DATABASE OPEN/MOUNT
- ALTER DATABASE BACKUP CONTROLFILE
- ALTER TABLESPACE BEGIN/END BACKUP
- ARCHIVE LOG
- RECOVER

The **OSDBA** role has the **OSOPER** role. In addition, the **OSDBA** role has the **WITH ADMIN OPTION** to allow it to grant system privileges to other users. This is the role that is used to create the database and for time-based recovery processes. Both the **OSOPER** and **OSDBA** roles include the **RESTRICTED SESSION** system privilege.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

If you intend to allow remote users to connect internal, you need to set the **REMOTE_LOGIN_PASSWORDFILE** option in your database parameter file to either **EXCLUSIVE** or **SHARED**. The user will then connect in server manager with the **AS SYSDBA** or **AS SYSOPER** clause at the end of the **CONNECT** command (**CONNECT SYS AS SYSDBA**). The privileges assigned to **SYSDBA** correspond to those for **OSDBA**. The privileges assigned to **SYSOPER** correspond to **OSOPER**. The operating system verifies the password provided using an external operating system file. This external file is generated using the **ORAPWD** utility. When the password for the **INTERNAL** or **SYS** accounts are changed with the **ALTER USER** command, the changes are mapped to the operating system password file.

CONNECT, RESOURCE, And DBA Roles

The CONNECT, RESOURCE, and DBA roles are predefined roles that are available for backward compatibility. These are created by Oracle when the database is created. When you create a user with Oracle Enterprise Manager, the CONNECT role is automatically granted to that user.

The following system privileges are granted to the CONNECT role:

- ALTER SESSION
- CREATE CLUSTER
- CREATE DATABASE LINK
- CREATE SEQUENCE
- CREATE SESSION
- CREATE SYNONYM
- CREATE TABLE
- CREATE VIEW

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

When you grant a user the RESOURCE role, that user is granted the **UNLIMITED TABLESPACE** system privilege as well. The following system privileges are granted to the RESOURCE role:

- CREATE CLUSTER
- CREATE PROCEDURE
- CREATE SEQUENCE
- CREATE TABLE
- CREATE TRIGGER
- CREATE TYPE

The DBA role includes all system privileges (95 separate grants), the capability to grant those system privileges to others, and **UNLIMITED TABLESPACE** is granted in the same way as with the **RESOURCE** role. If the **EXP_FULL_DATABASE** and **IMP_FULL_DATABASE** roles have been created, they are granted implicitly with the DBA role as are the **DELETE**, **EXECUTE** and **SELECT_CATALOG_ROLE** roles.

You can grant additional privileges to or revoke privileges from the **CONNECT**, **RESOURCE**, and **DBA** roles just as you would any other role that you created.

Export/Import Roles

Oracle provides a script entitled catexp.sql. This script creates the **EXP_FULL_DATABASE** and **IMP_FULL_DATABASE** roles. You can grant these to a user who will be executing the export and import utilities.

The **EXP_FULL_DATABASE** role has the **SELECT ANY TABLE** and **BACKUP ANY TABLE** system privileges. In addition, this role has **INSERT**, **DELETE**, and **UPDATE** privileges on the **SYS.INCVID**, **SYS.INCFIL**, and **SYS.INCEXP** tables.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The **IMP_FULL_DATABASE** role has the **BECOME USER** system privilege.

Using PROFILES

You can set up limits on the system resources used by setting up profiles with defined limits on resources. Profiles are very useful in large, complex organizations with many users. It allows you to regulate the amount of resources used by each database user by creating and assigning profiles to users. Using Oracle8 password attributes where added into profiles as well.

Creation of PROFILES

Profiles are a named set of resource limits. By default, when you create a user, they are given the default profile. The default profile provides unlimited use of all resources.

The syntax to create a profile follows:

```
>---CREATE PROFILE profile LIMIT
resource_parameters|password_parameters--;->
Resource_parameters (you can specify multiple paramters per
command):
 [SESSIONS_PER_USER n|UNLIMITED|DEFAULT]
 [CPU_PER_SESSION n|UNLIMITED|DEFAULT]
 [CPU_PER_CALL n|UNLIMITED|DEFAULT]
 [CONNECT_TIME              n|UNLIMITED|DEFAULT]
 [IDLE_TIME                 n|UNLIMITED|DEFAULT]
 [LOGICAL_READS_PER_SESSION n|UNLIMITED|DEFAULT]
 [LOGICAL_READS_PER_CALL    n|UNLIMITED|DEFAULT]
 [COMPOSITE_LIMIT           n|UNLIMITED|DEFAULT]
 [PRIVATE_SGA               n [K|M]|UNLIMITED|DEFAULT]
Password_parameters (Oracle8 and above):
 [FAILED_LOGIN_ATTEMPTS expr|UNLIMITED|DEFAULT]
 [PASSWORD_LIFE_TIME    expr|UNLIMITED|DEFAULT]
 [PASSWORD_REUSE_TIME   expr|UNLIMITED|DEFAULT]
 [PASSWORD_REUSE_MAX    expr|UNLIMITED|DEFAULT]
 [PASSWORD_LOCK_TIME    expr|UNLIMITED|DEFAULT]
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
[PASSWORD_GRACE_TIME   expr|UNLIMITED|DEFAULT]
[PASSWORD_VERIFY_FUNCTION function_name|NULL|DEFAULT]
```

Restrictions on password parameters:

- Expr must resolve to either an integer value or an integer number of days
- If PASSWORD_REUSE_TIME is set to an integer value, PASSWORD_REUSE_MAX must be set to UNLIMITED.
- If PASSWORD_REUSE_MAX is set to an integer value, PASSWORD_REUSE_TIME must be set to UNLIMITED.
- If both PASSWORD_REUSE_TIME and PASSWORD_REUSE_MAX are set to UNLIMITED, then Oracle uses neither of these password resources.
- If PASSWORD_REUSE_MAX is set to DEFAULT and PASSWORD_REUSE_TIME is set to UNLIMITED, then Oracle uses the PASSWORD_REUSE_MAX value defined in the DEFAULT profile.
- If PASSWORD_REUSE_TIME is set to DEFAULT and PASSWORD_REUSE_MAX is set to UNLIMITED, then Oracle uses the PASSWORD_REUSE_TIME value defined in the DEFAULT profile.
- If both PASSWORD_REUSE_TIME and PASSWORD_REUSE_MAX are set to DEFAULT, then Oracle uses whichever value is defined in the DEFAULT profile.

For example:

```
CREATE PROFILE  enduser  LIMIT
CPU_PER_SESSION                 60000
LOGICAL_READS_PER_SESSION   1000
CONNECT_TIME                30
PRIVATE_SGA                 102400
CPU_PER_CALL                UNLIMITED
COMPOSITE LIMIT             60000000
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
FAILED_LOGIN_ATTEMPTS       3
PASSWORD_LIFE_TIME              90
PASSWORD_REUSE_TIME             180
PASSWORD_LOCK_TIME              3
PASSWORD_GRACE_TIME             3
Verify_function_one ;
```

You can assign a profile to a user when you create the user or by altering the user. The syntax to alter the profile for a user is:

```
ALTER USER PROFILE profile;
```

For example:

```
ALTER USER scott
PROFILE appuser;
```

You must have the **CREATE PROFILE** system privilege to create a profile. To alter a profile you must be the creator of the profile or have the **ALTER PROFILE** system privilege. To assign a profile to a user, you must have the **CREATE USER** or **ALTER USER** system privilege.

Profiles and Resource Limits

The default cost assigned to a resource is unlimited. By setting resource limits, you can prevent users from performing operations that will tie up the system and prevent other users from performing operations. You can use resource limits for security to ensure that users log off the system and do not leave the session connected for long periods of time. You can also assign a composite cost to each profile . The system resource limits can be enforced at the session level, the call level or both.

The session level is from the time the user logs into the database until the user exits. The call level is for each SQL command issued. Session level limits are enforced for each connection. When a session level limit is exceeded, only the last SQL command

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

issued is rolled back and no further work can be performed until a commit, rollback or exit is performed. Table 7.15 lists the system resources which can be regulated at the session level.

One thing to note if you use parallel query option (PQO) is that the resources are applied to each new session, not accumulated over all of the sessions that a parallel operation uses.

SYSTEM RESOURCE	DEFINITION
CPU_PER_SESSION	total CPU time in hundreds of seconds
SESSIONS_PER_USER	number of concurrent sessions for a user
CONNECT_TIME	allowed connection time in minutes
IDLE_TIME	inactive time on the server in minutes
LOGICAL_READS_PER_SESSION	number of data blocks read including both physical and logical reads from memory and disk
PRIVATE_SGA	bytes of SGA used in a database with the multithreaded server (in K or M)

Table 7.15: Resources regulated at the session level

You can combine the CPU_PER_SESSION, LOGICAL_READS_PER_SESSION, CONNECT_TIME, and PRIVATE_SGA to create a COMPOSITE LIMIT.

Call-level limits are enforced during the execution of each SQL statement. When a call-level limit is exceeded, the last SQL command issued is rolled back. All the previous statements issued are still valid and the user can continue to execute other SQL

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

statements. The following system resources can be regulated at the call level:

- CPU_PER_CALL for the CPU time for the SQL statement
- LOGICAL_READS_PER_CALL for the number of data blocks read for the SQL statement

The assignment of a cost to a resource can be performed with the ALTER RESOURCE COST command. Resource limits that you set explicitly for a user take precedence over the resource costs in an assigned profile. The command line syntax for this command is:

```
>-ALTER RESOURCE COST ------------------------------------
--------------;-->
                        |-[CPU_PER_SESSION
n|UNLIMITED|DEFAULT]-----------|
                        |-[CONNECT_TIME n|UNLIMITED|DEFAULT]-
-------------|
                        |-[LOGICAL_READS_PER_SESSION
n|UNLIMITED|DEFAULT]-|
                        |-[PRIVATE_SGA  n
[K|M]|UNLIMITED|DEFAULT]--------|
```

For example,

```
ALTER RESOURCE COST CONNECT_TIME 100;
```

Use of resource limits is set in the database initialization parameter RESOURCE_LIMIT=TRUE. By default this parameter is set to false. This parameter can be changed interactively with an ALTER SYSTEM command.

The DBA_PROFILES view provides information on all the profiles and the resource limits for each profile. The RESOURCE_COST view shows the unit cost associated with each resource. Each user can find information on his resources and limits in the USER_RESOURCE_LIMITS view.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Table 7.16 gives a description of these data dictionary views.

DBA_PROFILES	
Column	Definition
Profile	the name given to the profile
Resource_name	the name of the resource assigned to the profile
Limit	the limit placed on the profile
RESOURCE_COST	
Column	Definition
Resource_name	name of the resource
Unit_cost	cost assigned
USER_RESOURCE_LIMITS	
Column	Definition
Resource_name	the name of the resource
Limit	the limit placed on the user

Table 7.16: Data dictionary views for resources.

Altering Profiles

Provided you have the CREATE PROFILE or ALTER PROFILE system privilege, you can alter any profile, including the Oracle created DEFAULT profile. You can alter a profile to change the cost assigned to each resource. The syntax to alter a profile follows:

```
>--ALTER PROFILE profile LIMIT
resource_parameters|password_parameters-;->

Resource_parameters:

 [SESSIONS_PER_USER n|UNLIMITED|DEFAULT]
 [CPU_PER_SESSION   n|UNLIMITED|DEFAULT]
 [CPU_PER_CALL      n|UNLIMITED|DEFAULT]
 [CONNECT_TIME      n|UNLIMITED|DEFAULT]
 [IDLE_TIME         n|UNLIMITED|DEFAULT]
 [LOGICAL_READS_PER_SESSION n|UNLIMITED|DEFAULT]
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
 [LOGICAL_READS_PER_CALL    n|UNLIMITED|DEFAULT]
 [COMPOSITE_LIMIT           n|UNLIMITED|DEFAULT]
 [PRIVATE_SGA               n [K|M|UNLIMITED|DEFAULT]]

Password_parameters (Oracle8 and above):

 [FAILED_LOGIN_ATTEMPTS expr|UNLIMITED|DEFAULT]
 [PASSWORD_LIFE_TIME    expr|UNLIMITED|DEFAULT]
 [PASSWORD_REUSE_TIME   expr|UNLIMITED|DEFAULT]
 [PASSWORD_REUSE_MAX    expr|UNLIMITED|DEFAULT]
 [PASSWORD_LOCK_TIME    expr|UNLIMITED|DEFAULT]
 [PASSWORD_GRACE_TIME   expr|UNLIMITED|DEFAULT]
 [PASSWORD_VERIFY_FUNCTION function_name|NULL|DEFAULT]
```

For example:

```
ALTER PROFILE enduser LIMIT
CPU_PER_SESSION 60000
LOGICAL_READS_PER_SESSION 1000
CONNECT_TIME 60
PRIVATE_SGA 102400
CPU_PER_CALL UNLIMITED
COMPOSITE LIMIT 60000000;
```

To disable a profile during a session, you must have the ALTER SYSTEM privilege. A limit that you set for the session overrides the previous limit set by the profile. To reset the profile to the limit originally set by the database, set the limit to DEFAULT.

```
ALTER SYSTEM SET RESOURCE_LIMIT =  TRUE|FALSE;
```

For example:

```
ALTER SYSTEM SET RESOURCE_LIMIT = TRUE ;
```

Profiles and Passwords

In Oracle8 the ability to control password expiry and password complexity and validity was added to Oracle. The capability to control passwords is contained within the purview of the profile.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Table 7.17 lists the password control attributes in a profile and their definitions.

Attribute	Description
FAILED_LOGIN_ATTEMPTS	specifies the number of failed attempts to log in to the user account before the account is locked.
PASSWORD_LIFE_TIME	limits the number of days the same password can be used for authentication. The password expires if it is not changed within this period, and further connections are rejected.
PASSWORD_REUSE_TIME	specifies the number of days before which a password cannot be reused. If you set PASSWORD_REUSE_TIME to an integer value, then you must set PASSWORD_REUSE_MAX to UNLIMITED.
PASSWORD_REUSE_MAX	specifies the number of password changes required before the current password can be reused. If you set PASSWORD_REUSE_MAX to an integer value, then you must set PASSWORD_REUSE_TIME to UNLIMITED.
PASSWORD_LOCK_TIME	specifies the number of days an account will be locked after the specified number of consecutive failed login attempts.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Attribute	Description
PASSWORD_GRACE_TIME	specifies the number of days after the grace period begins during which a warning is issued and login is allowed. If the password is not changed during the grace period, the password expires.
PASSWORD_VERIFY_ FUNCTION	allows a PL/SQL password complexity verification script to be passed as an argument to the CREATE PROFILE statement. Oracle provides a default script, but you can create your own routine or use third-party software instead. function is the name of the password complexity verification routine. NULL indicates that no password verification is performed.

Table 7.17: Password Control Attributes in a Profile

Oracle also provides a template PL/SQL procedure for use in creating your own password complexity and verification function. The example PL/SQL procedure is located in $ORACLE_HOME/rdbms/admin/utlpwdmg.sql on UNIX and in x:\orant\rdbms\admin\utlpwdmg.sql on NT. Other than the required input and return variables the password verification function can be as simple, or as complex, as you desire.

Managing CPU Utilization for in Oracle8i

In Oracle diverse groups of users may look at the database to find information important to their group. While we like to believe everyone is equal, face it, if the CEO wants a report his needs

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

come over and above Joe Clerk's needs for a different report. In earlier releases of Oracle you could use profiles to restrict specific types of resources but this was unwieldy and produced unpredictable results. New in Oracle8i is the concept of Oracle resource groups. A resource group specification allows you to specify that a specific group of database users can only use a certain percentage of the CPU resources on the system. A resource plan must be developed that defines the various levels within the application and their percentage of CPU resources in a waterfall type structure where each subsequent levels percentages are based on the previous levels.

Creating a Resource Plan

Rather than have a simple CREATE RESOURCE PLAN command, Oracle8i has a series of packages which must be run in a specific order to create a proper resource plan. All resource plans are created in a pending area before being validated and committed to the database. The requirements for a valid resource plan are outlined in the definition of the DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA procedure below.

Resource plans can have up to 32 levels with 32 groups per level allowing the most complex resource plan to be easily grouped. Multiple plans, sub-plans and groups can all be tied together into an application spanning CPU resource utilization rule set. This rule set is known as a set of directives. An example would be a simple 2-tier plan like that shown in Figure 1.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

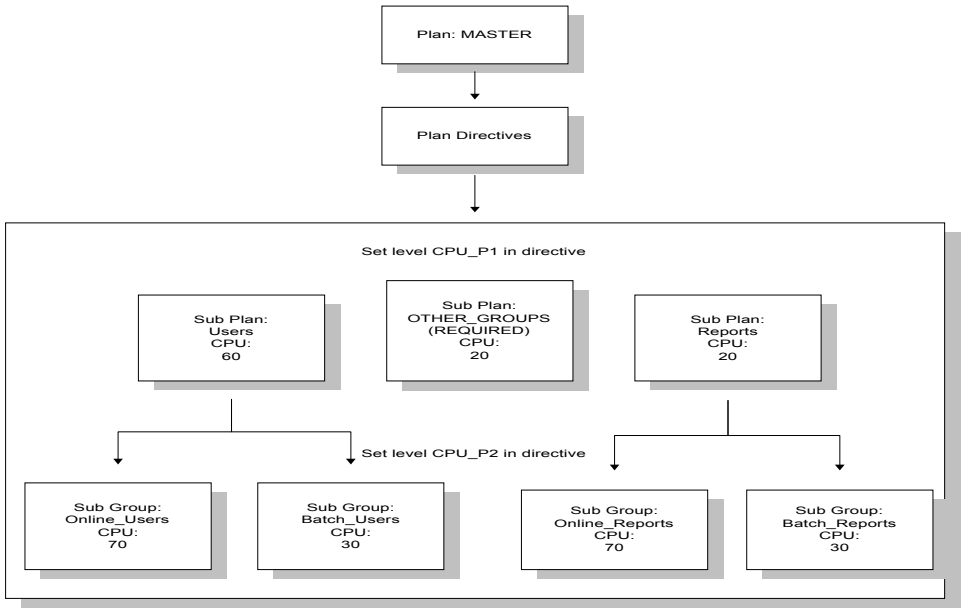


Figure 1 Example Resource Plan

An example of how this portioning out of CPU resources works would be to examine what happens in the plan shown in Figure 1. In figure 1 we have a top level called MASTER which can have up to 100% of the CPU if it requires it. The next level of the plan creates two sub-plans, USERS and REPORTS which will get maximums of 60 and 20 percent of the CPU respectively (we also have the required plan OTHER_GROUPS to which we have assigned 20 percent, if a user is not assigned to a specific group, they get OTHERS). Under USERS we have two groups, ONLINE_USERS and BATCH_USERS.

ONLINE_USERS gets 70 percent of USERS 60 percent or an overall percent of CPU of 42 percent while the other sub-group, BATCH_USERS gets 30 percent of the 60 percent for a total overall percent of 18.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The steps for creating a resource plan, its directives and its groups is shown in Figure 2.

One thing to notice about Figure 2 is that the last step shows several possible packages which can be run to assign or change the assignment of resource groups. The first package listed, DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_CON SUMER_GROUP must be run the first time a user is assigned to a resource group or you won't be able to assign the user to the group. After the user has been given the SWITCH_CONSUMER_GROUP system privilege you don't have to re-run the package.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

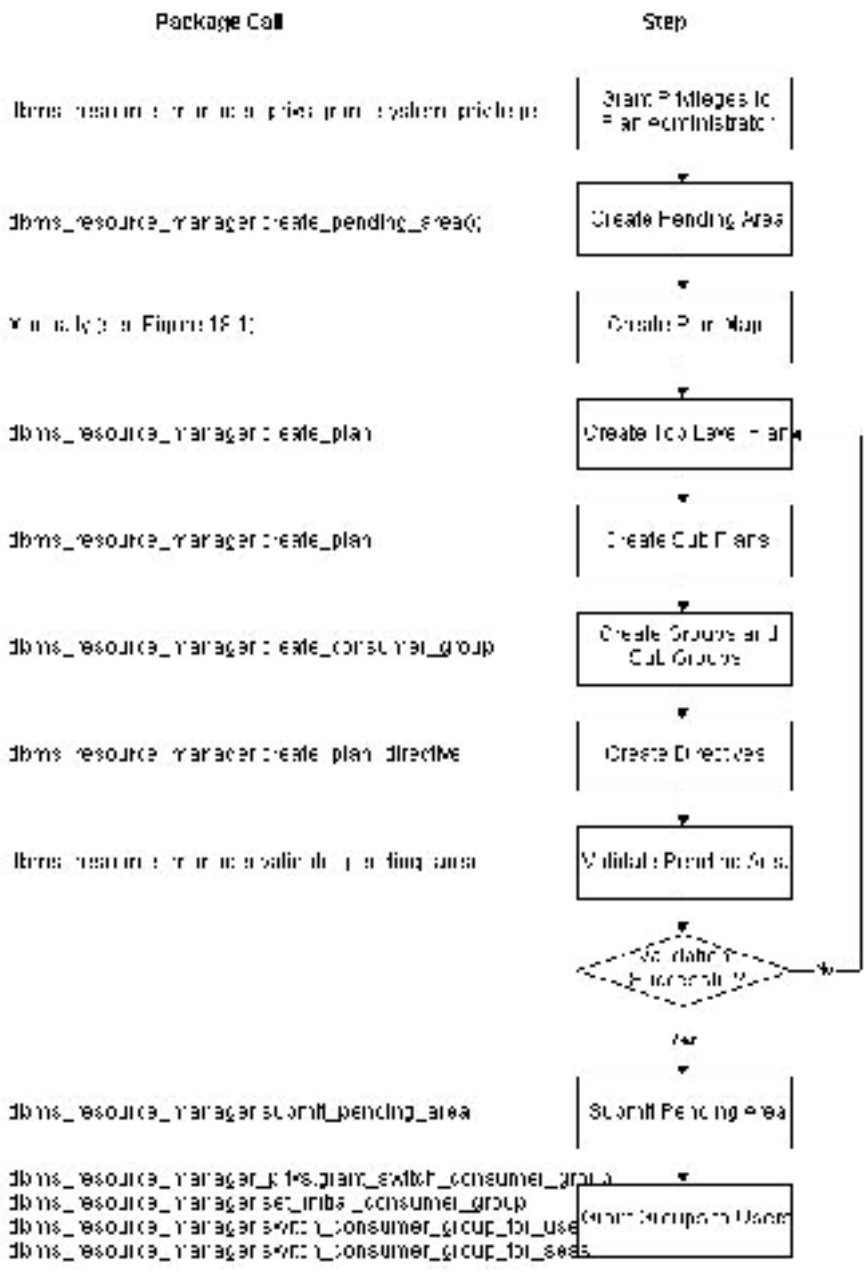


Figure 2 Steps to Create a Resource Plan

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Figure 3 shows the code to create the resource plan in Figure 1.

Figure 4 shows the results from running the source in figure 3.

```
set echo on
spool test_resource_plan.doc
-- Grant system privilege to plan administrator
--
execute
dbms_resource_manager_privs.grant_system_privilege('SYSTEM'
,'ADMINISTER_RESOURCE_MANAGER',TRUE);
--
--connect to plan administrator
--
CONNECT system/system_test@ortest1.world
--
-- Create Plan Pending Area
--
EXECUTE dbms_resource_manager.create_pending_area();
--
-- Create plan
--
execute dbms_resource_manager.create_plan('MASTER','Example
Resource Plan','EMPHASIS');
execute dbms_resource_manager.create_plan('USERS','Example
Resource Sub Plan','EMPHASIS');
execute
dbms_resource_manager.create_plan('REPORTS','Example
Resource Sub Plan','EMPHASIS');
--
--Create tiers of groups in plan
--
EXECUTE
dbms_resource_manager.create_consumer_group('ONLINE_USERS',
'3rd level group','ROUND-ROBIN');
EXECUTE
dbms_resource_manager.create_consumer_group('BATCH_USERS','
3rd level group','ROUND-ROBIN');
EXECUTE
dbms_resource_manager.create_consumer_group('ONLINE_REPORTS
','2rd level group','ROUND-ROBIN');
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
EXECUTE
dbms_resource_manager.create_consumer_group('BATCH_REPORTS'
,'2rd level group','ROUND-ROBIN');
--
-- Create plan directives
--
EXECUTE
dbms_resource_manager.create_plan_directive('MASTER',
'USERS',         0,60,0,0,0,0,0,0,NULL);
EXECUTE
dbms_resource_manager.create_plan_directive('MASTER',
'REPORTS',       0,20,0,0,0,0,0,0,NULL);
EXECUTE
dbms_resource_manager.create_plan_directive('MASTER','OTHER
_GROUPS',   0,20,0,0,0,0,0,0,NULL);
EXECUTE
dbms_resource_manager.create_plan_directive('USERS',
'ONLINE_USERS',  0,0,70,0,0,0,0,0,NULL);
EXECUTE
dbms_resource_manager.create_plan_directive('USERS',
'BATCH_USERS',   0,0,30,0,0,0,0,0,NULL);
EXECUTE
dbms_resource_manager.create_plan_directive('REPORTS','ONLI
NE_REPORTS',0,0,70,0,0,0,0,0,NULL);
EXECUTE
dbms_resource_manager.create_plan_directive('REPORTS','BATC
H_REPORTS', 0,0,30,0,0,0,0,0,NULL);
--
-- Verify Plan
--
EXECUTE dbms_resource_manager.validate_pending_area;
--
-- Submit Plan
--
EXECUTE dbms_resource_manager.submit_pending_area;
spool off
set echo off
```

Figure 3 Script to create example resource plan

Notice how the script in figure 3 follows the chart in Figure 2. These are the proper steps to create a resource plan. Figure 4 shows the results from running the script in Figure 3.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
SQL> -- Grant system privilege to plan administrator
SQL> --
SQL> execute
dbms_resource_manager_privs.grant_system_privilege('SYSTEM'
,'ADMINISTER_RESOURCE_MANAGER',TRUE);

PL/SQL procedure successfully completed.

SQL> --
SQL> --connect to plan administrator
SQL> --
SQL> CONNECT system/system_test@ortest1.world
Connected.
SQL> --
SQL> -- Create Plan Pending Area
SQL> --
SQL> EXECUTE dbms_resource_manager.create_pending_area();

PL/SQL procedure successfully completed.

SQL> --
SQL> -- Create plan
SQL> --
SQL> execute
dbms_resource_manager.create_plan('MASTER','Example
Resource Plan','EMPHASIS');

PL/SQL procedure successfully completed.

SQL> execute
dbms_resource_manager.create_plan('USERS','Example Resource
Sub Plan','EMPHASIS');

PL/SQL procedure successfully completed.

SQL> execute
dbms_resource_manager.create_plan('REPORTS','Example
Resource Sub Plan','EMPHASIS');

PL/SQL procedure successfully completed.

SQL> --
SQL> --Create tiers of groups in plan
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
SQL> --
SQL> EXECUTE
dbms_resource_manager.create_consumer_group('ONLINE_USERS',
'3rd level group','ROUND-ROBIN');

PL/SQL procedure successfully completed.

SQL> EXECUTE
dbms_resource_manager.create_consumer_group('BATCH_USERS','
3rd level group','ROUND-ROBIN');

PL/SQL procedure successfully completed.

SQL> EXECUTE
dbms_resource_manager.create_consumer_group('ONLINE_REPORTS
','2rd level group','ROUND-ROBIN');

PL/SQL procedure successfully completed.

SQL> EXECUTE
dbms_resource_manager.create_consumer_group('BATCH_REPORTS'
,'2rd level group','ROUND-ROBIN');

PL/SQL procedure successfully completed.

SQL> --
SQL> -- Create plan directives
SQL> --
SQL> EXECUTE
dbms_resource_manager.create_plan_directive('MASTER',
'USERS',        0,60,0,0,0,0,0,0,NULL);

PL/SQL procedure successfully completed.

SQL> EXECUTE
dbms_resource_manager.create_plan_directive('MASTER',
'REPORTS',      0,20,0,0,0,0,0,0,NULL);

PL/SQL procedure successfully completed.

SQL> EXECUTE
dbms_resource_manager.create_plan_directive('MASTER','OTHER
_GROUPS',   0,20,0,0,0,0,0,0,NULL);
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
PL/SQL procedure successfully completed.

SQL> EXECUTE
dbms_resource_manager.create_plan_directive('USERS',
'ONLINE_USERS',  0,0,70,0,0,0,0,0,NULL);

PL/SQL procedure successfully completed.

SQL> EXECUTE
dbms_resource_manager.create_plan_directive('USERS',
'BATCH_USERS',   0,0,30,0,0,0,0,0,NULL);

PL/SQL procedure successfully completed.

SQL> EXECUTE
dbms_resource_manager.create_plan_directive('REPORTS','ONLI
NE_REPORTS',0,0,70,0,0,0,0,0,NULL);

PL/SQL procedure successfully completed.

SQL> EXECUTE
dbms_resource_manager.create_plan_directive('REPORTS','BATC
H_REPORTS', 0,0,30,0,0,0,0,0,NULL);

PL/SQL procedure successfully completed.

SQL> --
SQL> -- Verify Plan
SQL> --
SQL> EXECUTE dbms_resource_manager.validate_pending_area;

PL/SQL procedure successfully completed.

SQL> --
SQL> -- Submit Plan
SQL> --
SQL> EXECUTE dbms_resource_manager.submit_pending_area;

PL/SQL procedure successfully completed.

SQL> spool off
```

Figure 4 Example run of script to create example resource plan

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

The other operations allowed against the components of the resource plan are alter and drop. Let's look at a quick drop example in Figure 5.

```
EXECUTE dbms_resource_manager.delete_plan('MASTER');
EXECUTE dbms_resource_manager.delete_plan('USERS');
EXECUTE dbms_resource_manager.delete_plan('REPORTS');
--
--delete tiers of groups in plan
--
EXECUTE
dbms_resource_manager.delete_consumer_group('ONLINE_USERS')
;
EXECUTE
dbms_resource_manager.delete_consumer_group('BATCH_USERS');
EXECUTE
dbms_resource_manager.delete_consumer_group('ONLINE_REPORTS
');
EXECUTE
dbms_resource_manager.delete_consumer_group('BATCH_REPORTS'
);
```

Figure 5 Example Drop Procedure

Notice how you must drop all parts of the plan, this is because Oracle allows Orphan groups and plans to exist. As you can tell from looking at the scripts the DBMS_RESOURCE_MANAGER and DBMS_RESOURCE_MANAGER_PRIVS packages are critical to implementing Oracle resource groups.

Let's examine these packages.

DBMS_RESOURCE_MANAGER Package

The DBMS_RESOURCE_MANAGER package is used to administer the new resource plan and consumer group options in Oracle8i. The package contains several procedures that are used to create, modify, drop and grant access to resource plans, groups, directives and pending areas. The invoker must have the

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

ADMINISTER_RESOURCE_MANAGER system privilege to execute these procedures. The procedures to grant and revoke this privilege are in the package DBMS_RESOURCE_MANAGER_PRIVS. The procedures in DBMS_RESOURCE_MANAGER are listed in table 1.

Procedure	Purpose
CREATE_PLAN	Creates entries which define resource plans.
UPDATE_PLAN	Updates entries which define resource plans.
DELETE_PLAN	Deletes the specified plan as well as all the plan directives it refers to.
DELETE_PLAN_CASCADE	Deletes the specified plan as well as all its descendants (plan directives, subplans, consumer groups).
CREATE_CONSUMER_GROUP	Creates entries which define resource consumer groups.
UPDATE_CONSUMER_GROUP	Updates entries which define resource consumer groups.
DELETE_CONSUMER_GROUP	Deletes entries which define resource consumer groups.
CREATE_PLAN_DIRECTIVE	Creates resource plan directives.
UPDATE_PLAN_DIRECTIVE	Updates resource plan directives.
DELETE_PLAN_DIRECTIVE	Deletes resource plan directives.
CREATE_PENDING_AREA	Creates a work area for changes to resource manager objects.
VALIDATE_PENDING_AREA	Validates pending changes for the resource manager.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Procedure	Purpose
CLEAR_PENDING_AREA	Clears the work area for the resource manager.
SUBMIT_PENDING_AREA	Submits pending changes for the resource manager.
SET_INITIAL_CONSUMER_GROUP	Assigns the initial resource consumer group for a user.
SWITCH_CONSUMER_GROUP_FOR_SESS	Changes the resource consumer group of a specific session.
SWITCH_CONSUMER_GROUP_FOR_USER	Changes the resource consumer group for all sessions with a given user name.

Table 1 DBMS_RESOURCE_MANAGER_PACKAGES

DBMS_RESOURCE_MANGER Procedure Syntax

The calling syntax for all of the DBMS_RESOURCE_MANAGER packages follow.

Syntax for the CREATE_PLAN Procedure:

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN (
   plan                       IN VARCHAR2,
   comment                    IN VARCHAR2,
   cpu_mth                    IN VARCHAR2 DEFAULT
'EMPHASIS',
   max_active_sess_target_mth IN VARCHAR2 DEFAULT

'MAX_ACTIVE_SESS_ABSOLUTE',
   parallel_degree_limit_mth  IN VARCHAR2 DEFAULT

'PARALLEL_DEGREE_LIMIT_ABSOLUTE');
```

Where:

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- Plan - the plan name
- Comment - any text comment you want associated with the plan name
- Cpu_mth - one of EMPHASIS or ROUND-ROBIN
- max_active_sess_target_mth - allocation method for max. active sessions
- parallel_degree_limit_mth - allocation method for degree of parallelism

Syntax for the UPDATE_PLAN Procedure:

```
DBMS_RESOURCE_MANAGER.UPDATE_PLAN (
   plan                            IN VARCHAR2,
   new_comment                     IN VARCHAR2 DEFAULT NULL,
   new_cpu_mth                     IN VARCHAR2 DEFAULT NULL,
   new_max_active_sess_target_mth IN VARCHAR2 DEFAULT NULL,
   new_parallel_degree_limit_mth  IN VARCHAR2 DEFAULT
NULL);
```

Where:

- plan - name of resource plan
- new_comment - new user's comment
- new_cpu_mth - name of new allocation method for CPU resources
- new_max_active_sess_target_mth - name of new method for max. active sessions
- new_parallel_degree_limit_mth - name of new method for degree of parallelism

Syntax for the DELETE_PLAN Procedure:

```
DBMS_RESOURCE_MANAGER.DELETE_PLAN (
   plan IN VARCHAR2);
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Where:

- Plan - Name of resource plan to delete.

Syntax for the DELETE_PLAN Procedure:

```
DBMS_RESOURCE_MANAGER.DELETE_PLAN_CASCADE (
   plan IN VARCHAR2);
```

Where:

- Plan - Name of plan.

Syntax for the CREATE_RESOURCE_GROUP Procedure:

```
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (
   consumer_group IN VARCHAR2,
   comment        IN VARCHAR2,
   cpu_mth        IN VARCHAR2 DEFAULT 'ROUND-ROBIN');
```

Where:

- consumer_group - Name of consumer group.
- Comment - User's comment.
- cpu_mth - Name of CPU resource allocation method.

Syntax for the UPDATE_RESOURCE_GROUP Procedure:

```
DBMS_RESOURCE_MANAGER.UPDATE_CONSUMER_GROUP (
   consumer_group IN VARCHAR2,
   new_comment    IN VARCHAR2 DEFAULT NULL,
   new_cpu_mth    IN VARCHAR2 DEFAULT NULL);
```

Where:

- plan - name of resource plan

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- new_comment - new user's comment
- new_cpu_mth - name of new allocation method for CPU resources
- new_max_active_sess_target_mth - name of new method for max. active sessions
- new_parallel_degree_limit_mth - name of new method for degree of parallelism

Syntax for the DELTE_RESOURCE_GROUP Procedure:

```
DBMS_RESOURCE_MANAGER.DELETE_CONSUMER_GROUP (
   consumer_group IN VARCHAR2);
```

Where:

- plan - name of resource plan.

Syntax for the CREATE_PLAN_DIRECTIVE Procedure:

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE (
   plan                      IN VARCHAR2,
   group_or_subplan          IN VARCHAR2,
   comment                   IN VARCHAR2,
   cpu_p1                    IN NUMBER   DEFAULT NULL,
   cpu_p2                    IN NUMBER   DEFAULT NULL,
   cpu_p3                    IN NUMBER   DEFAULT NULL,
   cpu_p4                    IN NUMBER   DEFAULT NULL,
   cpu_p5                    IN NUMBER   DEFAULT NULL,
   cpu_p6                    IN NUMBER   DEFAULT NULL,
   cpu_p7                    IN NUMBER   DEFAULT NULL,
   cpu_p8                    IN NUMBER   DEFAULT NULL,
   max_active_sess_target_p1 IN NUMBER   DEFAULT NULL,
   parallel_degree_limit_p1  IN NUMBER   DEFAULT NULL);
```

Where:

- plan - name of resource plan

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- group_or_subplan - name of consumer group or subplan
- comment - comment for the plan directive
- cpu_p1 - first parameter for the CPU resource allocation method
- cpu_p2 - second parameter for the CPU resource allocation method
- cpu_p3 - third parameter for the CPU resource allocation method
- cpu_p4 - fourth parameter for the CPU resource allocation method
- cpu_p5 - fifth parameter for the CPU resource allocation method
- cpu_p6 - sixth parameter for the CPU resource allocation method
- cpu_p7 - seventh parameter for the CPU resource allocation method
- cpu_p8 - eighth parameter for the CPU resource allocation method
- max_active_sess_target_p1 - first parameter for the max. active sessions allocation method
- (RESERVED FOR FUTURE USE)
- parallel_degree_limit_p1 - first parameter for the degree of parallelism allocation method

Syntax for the UPDATE_PLAN_DIRECTIVE Procedure:

```
DBMS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE (
   plan                             IN VARCHAR2,
   group_or_subplan                 IN VARCHAR2,
   new_comment                      IN VARCHAR2 DEFAULT NULL,
   new_cpu_p1                       IN NUMBER   DEFAULT NULL,
   new_cpu_p2                       IN NUMBER   DEFAULT NULL,
   new_cpu_p3                       IN NUMBER   DEFAULT NULL,
   new_cpu_p4                       IN NUMBER   DEFAULT NULL,
   new_cpu_p5                       IN NUMBER   DEFAULT NULL,
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
   new_cpu_p6                    IN NUMBER   DEFAULT NULL,
   new_cpu_p7                    IN NUMBER   DEFAULT NULL,
   new_cpu_p8                    IN NUMBER   DEFAULT NULL,
   new_max_active_sess_target_p1 IN NUMBER   DEFAULT NULL,
   new_parallel_degree_limit_p1  IN NUMBER   DEFAULT NULL);
```

Where:

- plan - name of resource plan
- group_or_subplan - name of group or subplan
- new_comment - comment for the plan directive
- new_cpu_p1 - first parameter for the CPU allocation method
- new_cpu_p2 - parameter for the CPU allocation method
- new_cpu_p3- parameter for the CPU allocation method
- new_cpu_p4 - parameter for the CPU allocation method
- new_cpu_p5 - parameter for the CPU allocation method
- new_cpu_p6 - parameter for the CPU allocation method
- new_cpu_p7 - parameter for the CPU allocation method
- new_cpu_p8 - parameter for the CPU allocation method
- new_max_active_sess_target_p1 - first parameter for the max. active sessions allocation method
- (RESERVED FOR FUTURE USE)
- new_parallel_degree_limit_p1 - first parameter for the degree of parallelism allocation method

Syntax for the DELETE_PLAN_DIRECTIVE Procedure:

```
DBMS_RESOURCE_MANAGER.DELETE_PLAN_DIRECTIVE (
   plan             IN VARCHAR2,
   group_or_subplan IN VARCHAR2);
```

Where:

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- plan - name of resource plan
- group_or_subplan - name of group or subplan.

Syntax for CREATE_PENDING_AREA Procedure:

This procedure lets you make changes to resource manager objects.

All changes to the plan schema must be done within a pending area. The pending area can be thought of as a "scratch" area for plan schema changes. The administrator creates this pending area, makes changes as necessary, possibly validates these changes, and only when the submit is completed do these changes become active.

You may, at any time while the pending area is active, view the current plan schema with your changes by selecting from the appropriate user views.

At any time, you may clear the pending area if you want to stop the current changes. You may also call the VALIDATE procedure to confirm whether the changes you has made are valid. You do not have to do your changes in a given order to maintain a consistent group of entries. These checks are also implicitly done when the pending area is submitted.

Note: Oracle allows "orphan" consumer groups (i.e., consumer groups that have no plan directives that refer to them). This is in anticipation that an administrator may want to create a consumer group that is not currently being used, but will be used in the future. The procedure has no arguments.

```
DBMS_RESOURCE_MANAGER.CREATE_PENDING_AREA;
```


COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Syntax of the VALIDATE_PENDING_AREA Procedure:

The VALIDATE_PENDING_AREA procedure is used to validate the contents of a pending area before they are submitted. The procedure has no arguments.

```
DBMS_RESOURCE_MANAGER.VALIDATE_PENDING_AREA;
```

Usage Notes For the Validate and Submit Procedures:

The following rules must be adhered to, and they are checked whenever the validate or submit procedures are executed:

- No plan schema may contain any loops.
- All plans and consumer groups referred to by plan directives must exist.
- All plans must have plan directives that refer to either plans or consumer groups.
- All percentages in any given level must not add up to greater than 100 for the emphasis resource allocation method.
- No plan may be deleted that is currently being used as a top plan by an active instance.
- For Oracle8i, the plan directive parameter, parallel_degree_limit_p1, may only appear in plan directives that refer to consumer groups (i.e., not at subplans).
- There cannot be more than 32 plan directives coming from any given plan (i.e., no plan can have more than 32 children).
- There cannot be more than 32 consumer groups in any active plan schema.
- Plans and consumer groups use the same namespace; therefore, no plan can have the same name as any consumer group.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- There must be a plan directive for OTHER_GROUPS somewhere in any active plan schema.This ensures that a session not covered by the currently active plan is allocated resources as specified by the OTHER_GROUPS directive.

If any of the above rules are broken when checked by the VALIDATE or SUBMIT procedures, then an informative error message is returned. You may then make changes to fix the problem(s) and reissue the validate or submit procedures.

Syntax of the CLEAR_PENDING_AREA Procedure:

The CLEAR_PENDING_AREA procedure clears the pending area without submitting it, all changes or entries are lost. The procedure has no arguments.

```
DBMS_RESOURCE_MANAGER.CLEAR_PENDING_AREA;
```

Syntax of the SUBMIT_PENDING_AREA Procedure:

The SUBMIT_PENDING_AREA procedure submits the contents of the pending area. First the contents are validated and then they are stored as valid in the database. The procedure has no arguments.

```
DBMS_RESOURCE_MANAGER.SUBMIT_PENDING_AREA;
```

Syntax of the SET_INITIAL_CONSUMER_GROUP Procedure:

The SET_INITIAL_CONSUMER_GROUP procedure sets the initial consumer group to which a user will belong. The user must have been granted SWITCH_RESOURCE_GROUP permission before you attempt to run this procedure.

```
DBMS_RESOURCE_MANAGER.SET_INITIAL_CONSUMER_GROUP (
   user           IN VARCHAR2,
   consumer_group IN VARCHAR2);
```

Where:

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- User – The user that is to have the resource group set.
- Consumer_group – The resource (or consumer) group to grant to the user.

Syntax of the SWITCH_CONSUMER_GROUP_FOR_SESS Procedure:

The SWITCH_RESOURCE_GROUP_FOR_SESS procedure allows an administrator to switch a user's consumer group for the duration of the current session.

```
DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_SESS(
  SESSION_ID      IN NUMBER,
  SESSION_SERIAL  IN NUMBER,
  CONSUMER_GROUP  IN VARCHAR2);
```

Where:

- session_id - SID column from the view V$SESSION
- session_serial - SERIAL# column from the view V$SESSION
- consumer_group - name of the consumer group of which to switch.

Syntax of the SWITCH_CONSUMER_GROUP_FOR_USER Procedure:

The SWITCH_CONSUMER_GROUP_FOR_USER switches a user's default consumer group to a new group. This is a permanent change.

```
DBMS_RESOURCE_MANAGER.SWITCH_CONSUMER_GROUP_FOR_USER (
   user           IN VARCHAR2,
   consumer_group IN VARCHAR2);
```

Where:

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- user - name of the user
- consumer_group - name of the consumer group to switch to

DBMS_RESOURCE_MANAGER_PRIVS Package

The DBMS_RESOURCE_MANAGER package has a companion package that grants privileges in the realm of the resource consumer option. The companion package is DBMS_RESOURCE_MANAGER_PRIVS. The procedures inside DBMS_RESOURCE_MANAGER_PRIVS are documented in table 2.

Procedure	Purpose
GRANT_SYSTEM_PRIVILEGE	Performs a grant of a system privilege.
REVOKE_SYSTEM_PRIVILEGE	Performs a revoke of a system privilege.
GRANT_SWITCH_CONSUMER_ GROUP	Grants the privilege to switch to resource consumer groups.
REVOKE_SWITCH_CONSUMER_ GROUP	Revokes the privilege to switch to resource consumer groups.

Table 2 DBMS_RESOURCE_MANAGER_PRIVS Procedures

DBMS_RESOURCE_MANGER_PRIVS Procedure Syntax

The calling syntax for all of the DBMS_RESOURCE_MANAGER_PRIVS packages follow.

Syntax for the GRANT_SYSTEM_PRIVILEGE Procedure:

The GRANT_SYSTEM_PRIVILEGE procedure grants ADMINISTER_RESOURCE_MANAGER privilege to a user. Currently there is only one resource group system grant.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SYSTEM_PRIVILEGE (
   grantee_name   IN VARCHAR2,
   privilege_name IN VARCHAR2 DEFAULT
'ADMINISTER_RESOURCE_MANAGER',
   admin_option   IN BOOLEAN);
```

Where:

- grantee_name - Name of the user or role to whom privilege is to be granted.
- privilege_name - Name of the privilege to be granted.
- admin_option - TRUE if the grant is with admin_option, FALSE otherwise.

Syntax for the REVOKE_SYSTEM_PRIVILGE Procedure:

The REVOKE_SYSTEM_PRIVILEGE procedure revokes the ADMINISTER_RESOURCE_MANAGER privilege from a user.

```
DBMS_RESOURCE_MANAGER_PRIVS.REVOKE_SYSTEM_PRIVILEGE (
   revokee_name   IN VARCHAR2,
   privilege_name IN VARCHAR2 DEFAULT
'ADMINISTER_RESOURCE_MANAGER');
```

Where:

- revokee_name - Name of the user or role from whom privilege is to be revoked.
- privilege_name - Name of the privilege to be revoked.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Syntax of the GRANT_SWITCH_CONSUMER_GROUP Procedure:

The GRANT_SWITCH_CONSUMER_GROUP procedure grants a user the ability to switch resource groups. This privilege must be granted to a user before their initial resource group can be granted.

```
DBMS_RESOURCE_MANAGER_PRIVS.GRANT_SWITCH_CONSUMER_GROUP (
   grantee_name   IN VARCHAR2,
   consumer_group IN VARCHAR2,
   grant_option   IN BOOLEAN);
```

Where:

- grantee_name - Name of the user or role to whom privilege is to be granted.
- consumer_group - Name of consumer group.
- grant_option - TRUE if grantee should be allowed to grant access, FALSE otherwise.

Usage Notes

- If you grant permission to switch to a particular consumer group to a user, then that user can immediately switch their current consumer group to the new consumer group.
- If you grant permission to switch to a particular consumer group to a role, then any users who have been granted that role and have enabled that role can immediately switch their current consumer group to the new consumer group.
- If you grant permission to switch to a particular consumer group to PUBLIC, then any user can switch to that consumer group.
- If the grant_option parameter is TRUE, then users granted switch privilege for the consumer group may also grant switch privileges for that consumer group to others.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- In order to set the initial consumer group of a user, you must grant the switch privilege for that group to the user.

Syntax of the REVOKE_SWITCH_CONSUMER_GROUP Procedure:

The REVOKE_SWITCH_CONSUMER_GROUP procedure revokes the ability of a user to switch their resource group.

```
DBMS_RESOURCE_MANAGER_PRIVS.REVOKE_SWITCH_CONSUMER_GROUP (
   revokee_name   IN VARCHAR2,
   consumer_group IN VARCHAR2);
```

Where:

- revokee_name - Name of user/role from which to revoke access.
- consumer_group - Name of consumer group.

Usage Notes

- If you revoke a user's switch privilege for a particular consumer group, then any subsequent attempts by that user to switch to that consumer group will fail.
- If you revoke the initial consumer group from a user, then that user will automatically be part of the DEFAULT_CONSUMER_GROUP (OTHERS) consumer group when logging in.
- If you revoke the switch privilege for a consumer group from a role, then any users who only had switch privilege for the consumer group via that role will not be subsequently able to switch to that consumer group.
- If you revoke the switch privilege for a consumer group from PUBLIC, then any users who could previously only use the

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

consumer group via PUBLIC will not be subsequently able to switch to that consumer group.

Summary

By carefully planning your resource allocation into plans and resource groups a multi-tier resource allocation plan can be quickly developed. By allocating CPU resources you can be sure that processing power is concentrated where it needs to be such that the CEO isn't waiting on a sub-clerk's process to finish before they get their results.

This lesson has shown how to use the various DBMS packages to configure and maintain a resource plan with its associated consumer groups.

This lesson is an excerpt from the upcoming book: "Oracle8i Administration and Management", Michael R. Ault, John Wiley and Sons publishing with permission.

Restricting Access by Rows in Oracle8i

New to Oracle8i is the concept of row level access restriction. For years DBAs have requested some form of conditional grant were access to specific rows can be easily restricted or granted based on user or group membership. Oracle has finally given DBAs the functionality of conditional grants in the form of row level security. In a data warehouse there may be data that is restricted in nature, the pay for a particular department, the locations of specific assets, etc. The new row level security, since it is restricted at the database level, prohibits access to restricted rows even when ad hoc tools are used to query the warehouse.

Row level security is managed using a combination of Oracle8i contexts, stored procedures, database level triggers and the DBMS_RLS package. The entire row level security concept is

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

tightly bound to the concept of a database policy. Generally speaking a policy will require:

- a context
- a procedure to implement the context
- a database (Oracle8i) level trigger that monitors login activity
- a security procedure to implement the policy
- a policy declaration

Row level security control depends on certain environment variables, know as contexts, to be set. The DBMS_CONTEXT package is used to set the various context variables used by the RLS policy.

Figure 1 shows a flowchart of how to implement a simple security policy.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

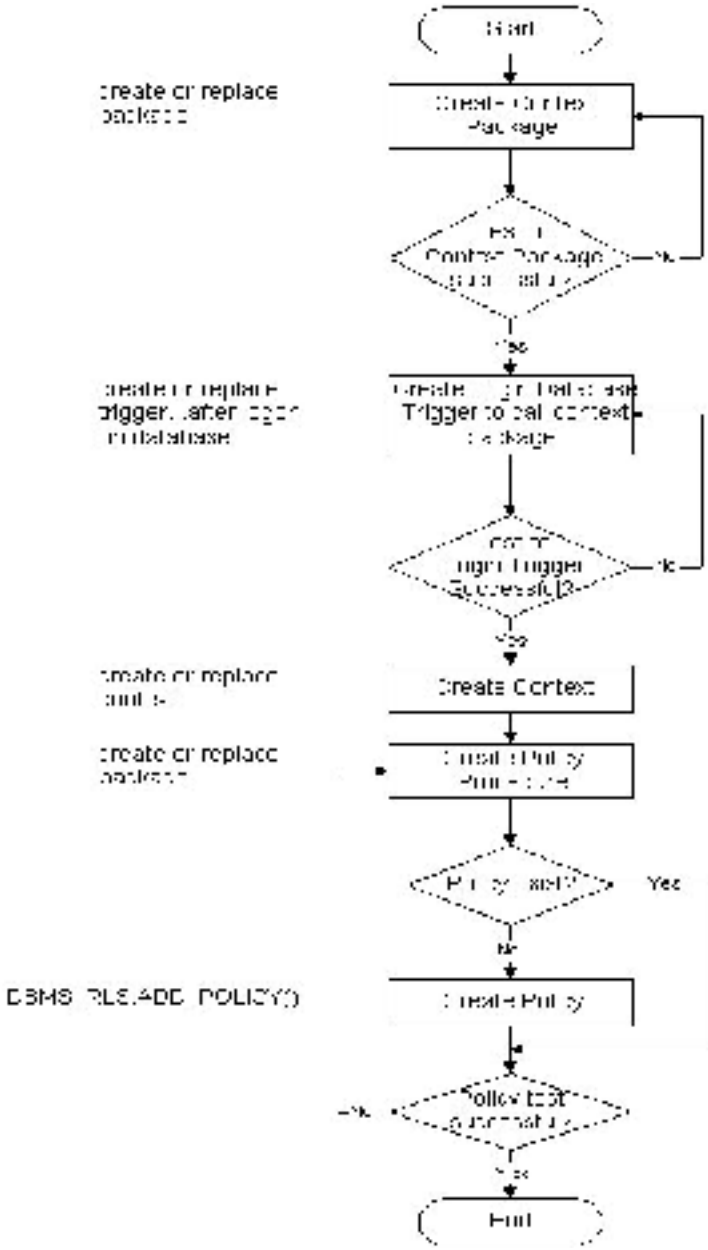


Figure 1: Steps to Implement a security policy

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

As you can see the process is not very complex. Let's examine each step and see what is really involved.

In the first step a context package or procedure is developed which will then be used by a login trigger to set each users context variables. This step is vital in that if the context variables aren't set it is many times more difficult to implement row level security using the DBMS_RLS package. The package or procedure used to set the context variables should resemble the one shown in figure 2.

```
CREATE OR REPLACE PACKAGE graphics_app AUTHID DEFINER AS
PROCEDURE get_graphics_function(usern IN VARCHAR2,
graphics_function OUT VARCHAR2);
PROCEDURE set_graphics_context(usern IN VARCHAR2);
END;
/
SET ARRAYSIZE 1
SHO ERR
CREATE OR REPLACE PACKAGE BODY graphics_app AS
graphics_user VARCHAR2(32);
graphics_function VARCHAR2(32);
PROCEDURE get_graphics_function(usern IN VARCHAR2,
graphics_function OUT VARCHAR2) IS
BEGIN
SELECT user_function INTO graphics_function FROM
graphics_dba.graphics_users
WHERE username=usern;
END get_graphics_function;
PROCEDURE set_graphics_context(usern IN VARCHAR2) IS
BEGIN
graphics_app.get_graphics_function(usern,graphics_function)
;
DBMS_SESSION.SET_CONTEXT('GRAPHICS_SEC','GRAPHICS_FUNCTION'
,graphics_function);
DBMS_SESSION.SET_CONTEXT('GRAPHICS_SEC','GRAPHICS_USER',use
rn);
END set_graphics_context;
END graphics_app;
/
SHOW ERR
```

Figure 2: Example Context Setting Procedure

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

In the package in figure 2 are two procedures, one that retrieves a users graphics function from a pre-built and populated table (GET_GRAPHICS_FUNCTION) and the other which is used to set the users context variables based on using the DBMS_SESSION.SET_CONTEXT procedure provided by Oracle (SET_GRAPHICS_CONTEXT).

Of course the procedures in figure 2 wouldn't be much use without a trigger that could run the procedure whenever a user logged on the system. Until Oracle8i this would have involved setting auditing on for login, moving the aud$ table from SYS ownership and setting the ownership to another user, resetting all of the synonyms pointing to aud$ and then building an on-insert trigger to perform the actual work. In Oracle8i all we have to do is build a database level trigger similar to the one shown in figure 3.

```
CREATE OR REPLACE TRIGGER set_graphics_context AFTER LOGON
ON DATABASE
DECLARE
username VARCHAR2(30);
BEGIN
  username:=SYS_CONTEXT('USERENV','SESSION_USER');
  graphics_app.set_graphics_context(username);
  EXCEPTION
   WHEN OTHERS THEN
   NULL;
END;
/
```

Figure 3: Example Database Logon Trigger

Once we have an operating context setting package and a database login trigger we can proceed to create the required context checking package and the context it checks. Figure 4 shows an example context checking package.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

```
CREATE OR REPLACE PACKAGE graphics_sec AUTHID DEFINER AS
FUNCTION graphics_check(obj_schema VARCHAR2, obj_name
VARCHAR2)
 RETURN VARCHAR2;
PRAGMA RESTRICT_REFERENCES(GRAPHICS_CHECK,WNDS);
END;
/
SET ARRAYSIZE 1
SHOW ERR
CREATE OR REPLACE PACKAGE BODY graphics_sec AS
FUNCTION graphics_check(obj_schema VARCHAR2, obj_name
VARCHAR2)
 RETURN VARCHAR2 AS
d_predicate VARCHAR2(2000);
user_context VARCHAR2(32);
BEGIN

user_context:=SYS_CONTEXT('graphics_sec','graphics_function
');
   IF user_context = 'ADMIN' THEN
     d_predicate:=' 1=1';
dbms_output.put_line(d_predicate);
   ELSIF user_context = 'GENERAL USER' THEN
     d_predicate:='
graphics_usage='||chr(39)||'UNRESTRICTED'||chr(39);
dbms_output.put_line(d_predicate);
   ELSIF user_context='DEVELOPER' THEN
     d_predicate:=' 1=1';
dbms_output.put_line(d_predicate);
   ELSIF user_context IS NULL THEN
     d_predicate:='1=2';
   END IF;
   RETURN d_predicate;
END graphics_check;
END;
/
SHOW ERR
```

Figure 4: Example Context Package

The entire purpose of the package in figure 4 is to return a d_predicate value based on a users graphics_function context value. The d_predicate value is appended to whatever WHERE

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

clause is included with their command, or is appended as a WHERE clause whenever there is no pre-existing clause.

The creation of our graphics security context is rather simple once we have finished the preliminary work, it boils down to one command:

```
CREATE OR REPLACE CONTEXT graphics_sec USING
sys.graphics_app;
```

The final step is to set the policy into the database. This is done with the DBMS_RLS package using the procedure ADD_POLICY:

```
BEGIN
  dbms_rls.add_policy(
  'GRAPHICS_DBA','INTERNAL_GRAPHICS','GRAPHICS_POLICY',
  'GRAPHICS_DBA','GRAPHICS_SEC.GRAPHICS_CHECK',
  'SELECT,INSERT,UPDATE,DELETE');
END;
```

The above policy simply ties the components we previously defined into a coherent entity called GRAPHICS_POLICY and implements this policy against the table INTERNAL_GRAPHICS which is in the schema GRAPHICS_DBA. The policy GRACPHICS_POLICY is owned by GRAPHICS_DBA and uses the procedure GRAPHICS_SEC.GRAPHICS_CHECK to verify users can perform SELECT, INSERT, UPDATE and DELETE operations.

The table graphics_users is required in the above example. The table contains the username and their graphics function.

Policy Usage

Policy usage is controlled internally by the Oracle system and adheres to the following usage guidelines:

- SYS user is not restricted by any security policy.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- The policy functions which generate dynamic predicates are called by the server. The following is the required structure for the function:

```
FUNCTION policy_function (object_schema IN VARCHAR2,
object_name VARCHAR2)
             RETURN VARCHAR2
```

  Where:

  - object_schema is the schema owning the table of view.
  - object_name is the name of table of view that the policy will apply.
- The maximum length of the predicate that the policy function can return is 2,000 bytes.
- The policy functions must have the purity level of WNDS (write no database state).
- Dynamic predicates generated out of different policies for the same object have the combined effect of a conjunction (ANDed) of all the predicates.
- The security check and object lookup are performed against the owner of the policy function for objects in the subqueries of the dynamic predicates.
- If the function returns a zero length predicate, then it is interpreted as no restriction being applied to the current user for the policy.
- When table alias is required (e.g., parent object is a type table) in the predicate, the name of the table or view itself must be used as the name of the alias. The server constructs the transient view as something like "select c1, c2, ... from tab where <predicate>".
- The checking of the validity of the function is done at runtime for ease of installation and other dependency issues import/export.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

DBMS_RLS Package

The entire concept of row level security is based on the use of policies stored in the database. The only way to store policies in the database is to use the DBMS_RLS package.

The DBMS_RLS procedures cause current DML transactions, if any, to commit before the operation. However, the procedures do not cause a commit first if they are inside a DDL event trigger. With DDL transactions, the DBMS_RLS procedures are part of the DDL transaction.

For example, you may create a trigger for CREATE TABLE. Inside the trigger, you may add a column through ALTER TABLE, and you can add a policy through DBMS_RLS. All these operations are in the same transaction as CREATE TABLE, even though each one is a DDL statement. The CREATE TABLE succeeds only if the trigger is completed successfully.

The DBMS_RLS package has the procedures shown in Table 1.

Procedure	Purpose
ADD_POLICY	Creates a fine-grained access control policy to a table or view.
DROP_POLICY	Drops a fine-grained access control policy from a table or view.
REFRESH_ POLICY	Causes all the cached statements associated with the policy to be re-parsed.
ENABLE_POLICY	Enables or disables a fine-grained access control policy.

Table 1 Procedure in DBMS_RLS Package

The syntax for calling the DBMS_RLS procedures are shown in the next sections.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

Syntax for the ADD_POLICY Procedure:

```
DBMS_RLS.ADD_POLICY (
   object_schema   IN VARCHAR2 := NULL,
   object_name     IN VARCHAR2,
   policy_name     IN VARCHAR2,
   function_schema IN VARCHAR2 := NULL,
   policy_function IN VARCHAR2,
   statement_types IN VARCHAR2 := NULL,
   update_check    IN BOOLEAN  := FALSE,
   enable          IN BOOLEAN  := TRUE);
```

Where:

- object_schema - schema owning the table/view, current user if NULL
- object_name - name of table or view
- policy_name - name of policy to be added
- function_schema - schema of the policy function, current user if NULL
- policy_function - function to generate predicates for this policy
- statement_types - statement type that the policy apply, default is any
- update_check - policy checked against updated or inserted value?
- enable - policy is enabled?

Syntax for the DROP_POLICY Procedure:

```
DBMS_RLS.DROP_POLICY (
   object_schema IN VARCHAR2 := NULL,
   object_name   IN VARCHAR2,
   policy_name   IN VARCHAR2);
```

Where:

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- object_schema - Schema containing the table or view (logon user if NULL).
- object_name - Name of table or view.
- policy_name - Name of policy to be dropped from the table or view.

Syntax for the REFRESH_POLICY Procedure:

```
DBMS_RLS.REFRESH_POLICY (
   object_schema IN VARCHAR2 := NULL,
   object_name   IN VARCHAR2 := NULL,
   policy_name   IN VARCHAR2 := NULL);
```

Where:

- object_schema - Schema containing the table or view.
- object_name - Name of table or view that the policy is associated with.
- policy_name - Name of policy to be refreshed.

Syntax for the ENABLE_POLICY Procedure:

```
DBMS_RLS.ENABLE_POLICY (
   object_schema IN VARCHAR2 := NULL,
   object_name   IN VARCHAR2,
   policy_name   IN VARCHAR2,
   enable        IN BOOLEAN);
```

Where:

- object_schema - Schema containing the table or view (logon user if NULL).
- object_name - Name of table or view that the policy is associated with.
- policy_name - Name of policy to be enabled or disabled.

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.

- Enable - TRUE to enable the policy, FALSE to disable the policy.

Through the use of the above procedures DBAs and developers can easily manage policies.

Summary

Oracle has given DBAs and developers a powerful new tool to manage row level security. This new tool is a combination of contexts, triggers and packages and a new package named DBMS_RLS through which security policies are implemented.

Through the proper use of policies, contexts, packages and database level triggers row level security can be easily integrated in Oracle8i applications.

(The above lesson was excerpted from the book: "Oracle8i Administration and Management", Michael R. Ault, John Wiley and Sons Publishers).

COPYRIGHT © 2003 RAMPANT TECHPRESS. ALL RIGHTS RESERVED.