

Oracle Rdb7™

Guide to Database Maintenance

Release 7.0

Part No. A41748-1

ORACLE®

Guide to Database Maintenance

Release 7.0

Part No. A41748-1

Copyright © 1984, 1996, Oracle Corporation. **All rights reserved.**

This software contains proprietary information of Oracle Corporation; it is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free.

Restricted Rights Legend Programs delivered subject to the DOD 'commercial computer software' and use, duplication and disclosure of the programs shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, programs delivered subject to the Federal Acquisition Regulations are 'restricted computer software' and use, duplication and disclosure of the programs shall be subject to the restrictions in FAR 52.227-14, Rights in Data—General, including Alternate III (Jun 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, back up, redundancy and other measures to ensure the safe use of such applications if the programs are used for such purposes, and Oracle disclaims liability for any damages caused by such use of the programs.

Oracle is a registered trademark of Oracle Corporation. Hot Standby, Oracle CDD/Administrator, Oracle CDD/Repository, Oracle CODASYL DBMS, Oracle Rally, Oracle Rdb, Oracle RMU, Oracle RMUwin, Oracle SQL/Services, and Rdb7 are trademarks of Oracle Corporation.

All other company or product names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

Contents

Send Us Your Comments	xix
Preface	xxi
Technical Changes and New Features	xxvii
1 Oracle Rdb Database Maintenance	
1.1 Database Administrator Responsibilities	1-1
1.2 Database Management Requirements	1-2
1.3 Database Maintenance Tools	1-3
1.4 Database Maintenance Tasks	1-4
1.5 Database Availability	1-8
1.5.1 Fault-Tolerant Oracle Rdb Databases	1-9
1.5.2 Online Backup Operations	1-9
1.5.3 Online By-Area and By-Page Restore and Recovery Operations, and By-Area Move Operations	1-10
1.5.4 Disabling Journaling for Write-Once Storage Areas	1-10
1.5.5 Cluster and Networkwide Automatic Recovery	1-10
1.5.6 Automatic Cleanup	1-10
1.5.7 Online DBA Activities	1-11
1.5.8 Offline DBA Activities	1-14
1.5.9 DBA Activities Requiring a Database Reload Operation	1-15
1.5.10 Quick and Automatic Database Recovery	1-15
1.5.11 Database Integrity	1-15
1.5.12 Checking Database Integrity and Evaluating Performance	1-16
1.6 Creating Sample Single-File and Multifile Databases	1-16

2 Monitoring Your Oracle Rdb Databases

2.1	The Oracle Rdb Monitor Process	2-2
2.1.1	Starting and Stopping the Monitor Process Interactively	2-3
2.1.2	Renaming the Monitor Log File	2-5
2.1.3	Changing the Monitor Process Priority	2-7
2.1.4	Reopening the Monitor Log File	2-7
2.1.5	Reading the Monitor Log File	2-8
2.2	Listing Active User Information	2-11
2.3	Displaying Clusterwide User Information	2-13
2.4	Displaying Database Characteristics	2-16

3 Security Auditing on OpenVMS

3.1	An Overview of Oracle Rdb Security Auditing	3-2
3.1.1	Default Security Auditing and Oracle RMU Security Auditing Commands	3-2
3.1.2	Use of OpenVMS Security Auditing	3-3
3.1.3	Monitoring Security Auditing Resources	3-4
3.2	Security Audit Event Types	3-4
3.2.1	The Audit Event Type	3-4
3.2.2	The Daccess Event Type	3-5
3.2.3	The Protection Event Type	3-6
3.2.4	The RMU Event Type	3-6
3.3	Defining Security Events to Be Audited	3-7
3.3.1	Setting User-Level Information for Security Auditing	3-9
3.3.2	Setting Audit Access to Database Objects (Daccess Auditing)	3-9
3.3.3	Enabling and Disabling Event Information for Security Auditing	3-12
3.3.4	Starting and Stopping Security Auditing and Other Auditing Characteristics	3-13
3.4	Reviewing Security Audit Information	3-16
3.4.1	Interpreting Security Auditing Alarm Information	3-17
3.4.1.1	Interpreting AUDIT Event Alarm Information	3-18
3.4.1.2	Interpreting Daccess Event Alarm Information	3-18
3.4.1.3	Interpreting Protection Event Alarm Information	3-20
3.4.1.4	Interpreting Oracle RMU Event Alarm Information	3-20
3.4.2	Using the RMU Load Audit Command	3-22
3.4.3	Reviewing Audit Journal Records	3-24

4 Opening and Closing a Database

4.1	Opening a Database	4-2
4.1.1	Using the RMU Open Command	4-3
4.1.2	Attaching to a Database	4-5
4.2	Closing a Database	4-7
4.2.1	Closing a Database and Using SQL	4-9
4.2.2	Using the Noabort Qualifier to Close a Database	4-13

5 Verifying the Integrity of Your Oracle Rdb Database

5.1	Why You Should Verify Your Database	5-1
5.2	Causes of Database Corruption	5-2
5.3	What Happens When You Verify Your Database	5-3
5.4	What the Full Verify Operation Checks	5-4
5.5	What Problems the Full Verify Operation Can Detect	5-7
5.6	Devising a Full Verify Strategy to Detect Problems	5-9
5.7	Interaction of RMU Verify Command Qualifiers	5-12
5.8	Measuring and Improving Verification Performance	5-16
5.9	Examples of Verify Operations on the mf_personnel Sample Database	5-17
5.10	Troubleshooting Suspected Problems	5-21
5.10.1	Using a Checksum Verification to Detect Page Corruption	5-23
5.10.2	Detecting a Data Integrity Corruption	5-25
5.10.3	Summary of RMU Verify Command Qualifiers for Troubleshooting	5-31
5.11	Examples of Database Corruption	5-33
5.11.1	Line Index Corruption	5-33
5.11.2	Logical Area Corruption	5-38
5.11.3	Data Integrity Corruption	5-43

6 Repairing or Altering a Database

6.1	Using RMU Repair	6-1
6.2	Using RdbALTER	6-2
6.2.1	Attaching to a Database	6-4
6.2.2	Clearing a Corruption Flag	6-5
6.2.3	Selecting the Area Page for Altering	6-6
6.2.4	Displaying Page Contents	6-10
6.2.5	Changing Page Contents	6-12
6.2.6	Moving Database Files	6-16
6.2.7	Moving Data	6-17

6.2.8	Using the RMU Alter Command to Remove References to .ruj Files	6-22
6.2.9	Clearing an Inconsistent Flag	6-23
6.2.10	Changing the Radix	6-25
6.2.11	Verifying Alterations	6-25
6.2.12	Keeping a Log or an Audit Trail of Alterations	6-25
6.2.13	Completing Transactions	6-26
6.2.14	Exiting from the RdbALTER Utility	6-27
6.2.15	Accessing Online Information	6-27

7 Backing Up Your Database

7.1	Introduction to Database Backup	7-1
7.2	How Does Oracle RMU Back Up a Database?	7-2
7.2.1	Oracle RMU Default Backup Operation	7-3
7.2.2	Oracle RMU Parallel Backup Operation	7-4
7.3	Invoking the RMU Backup Command	7-7
7.3.1	What Does Oracle RMU Back Up?	7-8
7.3.2	Writing the Backup Output File	7-9
7.3.2.1	Writing Output to Multiple Tapes	7-10
7.3.2.2	Output to Tape or Disk	7-11
7.3.2.3	Contents and Size	7-12
7.3.2.4	Protection	7-13
7.4	Why Not Use Other Backup Utilities?	7-13
7.5	Database Backup Strategies	7-15
7.5.1	Determining How Frequently You Need to Back Up Your Database	7-16
7.5.2	Requirements for Using a Full and Complete Backup Operation	7-17
7.5.3	Guidelines for Choosing a Full Versus an Incremental Backup Operation	7-18
7.5.4	Guidelines for Choosing a By-Area Backup Operation	7-19
7.5.4.1	Strategies	7-19
7.5.4.2	Command Qualifiers	7-22
7.5.5	Guidelines for Performing Online Versus Offline Backup Operations	7-22
7.5.6	Guidelines for Choosing Parallel Backup Operations	7-23
7.6	Implementing a Reliable Database Backup Procedure	7-25
7.6.1	Recommendations	7-25
7.6.2	Sample Backup Procedure	7-26
7.6.3	Computing Working Set Requirements for OpenVMS	7-29
7.6.4	Checking the Database Page Checksum During a Backup to Disk	7-30
7.7	Performing Full and Complete Database Backup Operations	7-32
7.8	Performing an Incremental Database Backup Operation	7-34

7.8.1	Starting an Incremental Backup Operation	7-35
7.8.2	Optimizing Incremental Backup Performance	7-37
7.8.3	Determining Which Pages Have Changed Since the Last Backup . . .	7-38
7.8.3.1	Looking at Timestamps	7-39
7.8.3.2	Checking By-Area Backup Timestamps	7-41
7.8.4	Measuring the Benefits of an Incremental Backup	7-41
7.9	Performing a By-Area Backup Operation	7-42
7.10	Performing a Parallel Backup Operation	7-45
7.10.1	Oracle RMU Commands for Parallel Backup Operations	7-46
7.10.2	Starting a Parallel Backup Operation	7-48
7.10.3	Parallel Backup Plan File	7-49
7.10.4	What Happens During a Parallel Backup Operation?	7-52
7.10.5	Using Loader Synchronization When Performing a Parallel Backup Operation	7-54
7.10.6	Monitoring the Progress of a Parallel Backup Operation	7-55
7.11	Performing Online Backup Operations	7-56
7.11.1	Avoiding Lock Conflicts	7-59
7.11.2	Setting Lock Timeout Intervals	7-60
7.12	Backing Up a Database to a Disk Device	7-61
7.13	Backing Up a Database to Tape Devices	7-62
7.13.1	Mounting One or More Tapes on a Single Tape Drive	7-63
7.13.2	Using Multiple Tape Drives	7-65
7.13.2.1	Using Concurrent Tape Drives Efficiently	7-67
7.13.2.2	Controlling Tape Concurrency with the Master Qualifier	7-68
7.13.2.3	Preloading Tapes Using Load Synchronization	7-69
7.13.2.4	Optimizing Tape Utilization Using a Journal File	7-71
7.13.3	Avoiding Underrun Errors Using Cyclic Redundancy Checks	7-71
7.13.4	Checking Tape Labels	7-73
7.13.5	Monitoring Error Rates	7-76
7.14	Displaying Database Backup Information	7-77

8 Restoring Your Database

8.1	Preparing to Restore a Database	8-1
8.2	Access Privileges for a Restored Database	8-2
8.3	Full Database Restore Operations	8-5
8.4	Incremental Restore Operations	8-10
8.5	A Sample Restore Procedure	8-13
8.6	Performing By-Area Restore Operations	8-14
8.6.1	Performing an Online By-Area Restore Operation on One Storage Area	8-18
8.7	Performing By-Page Restore Operations	8-18
8.8	Restoring Only the Root File from a Database Backup File	8-22

8.9	Restoring a Database Directly from Tape	8-32
8.9.1	Using a Single Tape Drive	8-32
8.9.2	Using Multiple Tape Drives	8-33
8.10	Exceeded Quotas During a Database Restore or Backup Operation	8-36
8.11	Modifying Database Characteristics During a Restore Operation	8-37
8.11.1	Modifying After-Image Journaling Characteristics	8-40
8.11.2	Modifying SPAM Thresholds Values	8-43
8.11.3	Modifying Blocks per Page	8-43
8.11.4	Restoring List Storage Areas to WORM Optical or Read/Write Disk Devices	8-44
8.12	Performing Additional Tasks During a Restore Operation	8-45
8.12.1	Using an Options File to Restore a Database	8-45
8.12.2	Creating a Duplicate Database During a Restore Operation	8-47
8.12.3	Moving Database Files	8-49
8.12.4	Moving and Updating Data Dictionary Information	8-50
8.13	Using the SQL EXPORT and IMPORT Statements	8-52

9 After-Image Journaling and Recovery

9.1	Introduction	9-1
9.1.1	Recommended and Required Usage	9-2
9.1.2	Information Written to the After-Image Journal File	9-3
9.1.3	Journaling Strategy	9-4
9.1.4	Displaying After-Image Journaling Performance	9-5
9.2	Enabling After-Image Journaling	9-5
9.3	Disabling After-Image Journaling	9-8
9.4	The After-Image Journal (.aij) File	9-8
9.4.1	Location and Accessibility	9-9
9.4.2	Extensible and Fixed-Size Journal Files	9-10
9.4.3	Setting the Allocation Size	9-12
9.4.4	Setting the Extent Size for an Extensible Journal File	9-14
9.5	Journaling List (WORM) Data	9-16
9.6	Backing Up After-image Journal Files	9-19
9.6.1	Backup and Recovery Strategy	9-19
9.6.2	Disk and Tape Backup Media	9-20
9.6.3	Reusing Disk and Tape Backup Media	9-21
9.6.4	Scheduling	9-22
9.6.5	Transaction Sequence Numbers (TSN)	9-22
9.6.6	Backup Termination or Failure	9-25
9.6.7	Backing Up a Single Extensible Journal File	9-25
9.6.8	Backing Up Multiple Fixed-Size Journal Files	9-25
9.6.8.1	Automatic and Manual Backup Operations	9-26
9.6.8.2	File Management	9-31

9.6.9	Writing a Customized Backup Procedure	9-32
9.7	Journal Switchover	9-35
9.7.1	Reasons Why After-Image Journal File Switchover Suspends	9-36
9.7.2	Avoiding Journal File Switchover Suspension	9-36
9.7.3	Determining If Journal File Switchover Is Suspended	9-43
9.7.4	Resuming After-Image Journaling Operations	9-44
9.7.4.1	Manually Performing a Full or By-Sequence After-Image Journal File Backup	9-45
9.7.4.2	Adding a New After-Image Journal File	9-46
9.7.5	Recovering the Database	9-50
9.8	Optimizing After-Image Journaling Performance	9-52
9.8.1	Fast Commit Processing	9-53
9.8.1.1	Changes to the Journal File	9-54
9.8.1.2	Effects On the Journal File Backup Operations	9-55
9.8.1.3	Disk Space Requirements for an Extensible Journal File	9-57
9.8.1.4	Effects When Fixed-Size Journal Files Switch Over	9-57
9.8.2	Commit To Journal Option	9-58
9.9	Recovering Transactions from Journal Files	9-58
9.9.1	Automatic Recovery for Fixed-Size Journal Files	9-59
9.9.2	Steps for Recovering a Database	9-59
9.9.3	Manually Recovering Journal Files	9-68
9.9.4	Optimizing Recovery Performance	9-73
9.10	What Causes an After-Image Journal File to Be Inaccessible	9-75
9.10.1	Recovering a Lost Extensible Journal File	9-76
9.10.2	Recovering a Lost Fixed-Size Journal File	9-77
9.11	Displaying the Contents of a Journal File	9-77
9.12	Example of Database Backup, Recover, and Restore Journaling Operations	9-82

10 Recovery-Unit Journaling and Recovery

10.1	The Recovery-Unit Journal File	10-1
10.1.1	Directory Location	10-2
10.1.2	Recovery for Update Transactions	10-4
10.2	Improving Performance of the Automatic Recovery Process	10-6
10.2.1	Setting the Number of Database Buffers	10-6
10.2.2	Adjusting the Number of Recovery Buffers	10-7
10.3	Displaying the Contents of an .ruj File	10-8

11 Displaying Root Files, Storage Areas, and Snapshot Files

11.1	Using the RMU Dump Command	11-1
11.2	Data Structures	11-3
11.2.1	Storage Areas with Uniform Page Format	11-3
11.2.2	Storage Areas with Mixed Page Format	11-6
11.3	Displaying Data Storage Files	11-7
11.4	Displaying Logical Areas	11-12
11.5	Displaying Snapshot Files	11-18
11.5.1	Snapshot Page Tail	11-22

12 Displaying the Contents of Data Storage Pages

12.1	Page Header for a Data Storage Page	12-3
12.2	Line Index for a Data Storage Page	12-4
12.3	TSN Index for a Data Storage Page	12-5
12.4	Locked and Unlocked Free Space for a Data Storage Page	12-6
12.5	Storage Segment Structure for a Data Storage Page	12-8
12.5.1	User-Stored Data Storage Segments	12-9
12.5.2	List Storage Segments	12-10
12.5.3	Index Node Storage Segments	12-18
12.5.3.1	Sorted Index Node Records	12-20
12.5.3.2	Hashed Index Node Records	12-25
12.6	Page Tail for a Data Storage Page	12-35
12.7	Fragmented Storage Records	12-36

13 Displaying the Contents of SPAM Pages

13.1	Space Management in Single-File and Multifile Databases	13-1
13.2	Space Management for Logical Areas	13-2
13.3	SPAM Pages in Storage Areas with Uniform Page Format	13-4
13.3.1	Area Bit Maps	13-8
13.3.2	Area Inventory Pages	13-10
13.4	SPAM Pages in Storage Areas with Mixed Page Format	13-12
13.5	SPAM Pages in Storage Areas with Mixed Page Format Without a Placement Index	13-17

A Handling Bugcheck Dumps

A.1	Submitting Problem Reports	A-1
A.2	Troubleshooting Oracle Rdb	A-1
A.2.1	Types of Bugcheck Dumps	A-2
A.2.2	Locations of Bugcheck Dump Files	A-3
A.2.3	Defining the RDMSBUGCHECK_DIR Logical Name	A-4
A.3	Understanding Error Messages and Bugcheck Dump Exceptions	A-5
A.3.1	The %RDMS-F-TERMINATE Error	A-5
A.3.2	Exceeding Quotas	A-7
A.3.2.1	Disk Quota Exceeded	A-7
A.3.2.2	Process Quota Exceeded	A-8
A.3.3	Using an Invalid dbkey in an Update Transaction	A-9
A.4	Reporting a Bugcheck Dump	A-9
A.4.1	Getting a Bugcheck Dump	A-10
A.4.2	Examining a Bugcheck Dump	A-10
A.4.3	Contents of a Bugcheck Dump	A-11

Index

Examples

2-1	Stopping the Monitor Process	2-3
2-2	Stopping the Monitor Process and Aborting User Processes	2-4
2-3	Starting the Monitor Process	2-5
2-4	Displaying the Contents of the Monitor Log File	2-5
2-5	Specifying a Different Device and Directory for the Monitor Log File	2-6
2-6	Displaying the Contents of the Monitor Log File	2-6
2-7	Stopping and Changing the Monitor Process Priority	2-7
2-8	Reopening the Monitor Log File	2-7
2-9	Showing Oracle Rdb System Status	2-8
2-10	Reopening the Monitor Log File and Showing Oracle Rdb System Status	2-8
2-11	Reading the Contents of the Monitor Log File	2-9
2-12	Using an Editor to Read the Contents of the Monitor Log File	2-10
2-13	Showing Oracle Rdb System Status	2-12
2-14	Showing the Current Version of Oracle Rdb	2-12
2-15	Showing Open Databases and Attached Users	2-13
2-16	Displaying a List of Database Users on a Cluster System	2-14

2-17	Using SYSMAN and the RMU Show Users Command on a Cluster Configuration	2-15
4-1	Opening the Database Manually and Showing Users	4-4
4-2	Closing the Database and Showing Users	4-4
4-3	Changing the Database Opening to Automatic and Showing Users	4-6
4-4	Closing Databases and Showing Users	4-9
4-5	Closing the Database	4-10
4-6	Opening and Then Closing the Database	4-11
4-7	Trying to Access a Closing Database	4-11
4-8	Changing Database Access to Manual Open	4-12
4-9	Closing the Database	4-12
4-10	Changing Database Access to Automatic Open	4-12
4-11	Closing the Database Through Attrition for a Specific Node	4-13
4-12	Showing Active Users Attached to the Database	4-13
4-13	Showing No Active Users Attached to the Database	4-14
4-14	Closing the Database Through Attrition for All Nodes	4-14
5-1	The Log File from a Full Verify Operation	5-17
5-2	Verifying the Integrity of the Database	5-20
5-3	Using Exclusive Access During a Verify Operation	5-20
5-4	Display of the Corrupt Page Table	5-22
5-5	Verifying Only Database Page Checksums	5-24
5-6	Verifying the Constraints for a Database	5-26
5-7	Page Checksum Bad Warning Message and GAPONPAGE Error Message Returned from a DEPARTMENTS Area Verify Operation	5-33
5-8	Online Help File Explanation for the RMU Verify GAPONPAGE Error Message	5-34
5-9	Corrupted Page 2 of the DEPARTMENTS Storage Area	5-35
5-10	Uncorrupted Page 2 of the DEPARTMENTS Storage Area	5-37
5-11	Page Checksum Bad Warning Message Returned from a Checksum Verify Operation of the DEPARTMENTS Logical Area	5-39
5-12	B-Tree Lexical Error Returned from an Index Verify Operation with No Data Record Check	5-40
5-13	B-Tree Lexical Error Returned from a Checksum Verify Operation with a Data Record Check	5-40
5-14	Corrupted Portion of Page 2 of the DEPARTMENTS Storage Area	5-42

5-15	Uncorrupted Portion of Page 2 of the DEPARTMENTS Storage Area	5-43
5-16	Page Errors Returned from an Area Verify Operation of the JOBS Storage Area	5-44
5-17	Fatal Error Messages Returned from a Constraint Verify Operation	5-44
5-18	Tracing the Corruption to a Set of Constraints	5-45
5-19	Corrupted Portion of Page 2 of the JOBS Storage Area	5-46
5-20	Verifying the JOBS Storage Area	5-47
5-21	Uncorrupted Portion of Page 2 of the JOBS Storage Area	5-47
6-1	RMU Dump Command to Display Contents of the EMPIDS_LOW Storage Area	6-8
6-2	Display of Line 1 of Page 2 in Area 2 (EMPIDS_LOW Storage Area)	6-10
6-3	Changing Data on Page 2 in the EMPIDS_LOW Storage Area	6-14
6-4	Page Verification Using the RdbALTER Utility Returns a Checksum Error	6-14
6-5	Depositing a New Checksum on Page 2 in the EMPIDS_LOW Storage Area	6-15
6-6	Checksum Error Returned from a Full Verification	6-18
6-7	Display of Page 3 of the EMP_INFO Storage Area	6-19
6-8	Moving Contents of Line 5 to a Designated Location, Cleaning Up Extraneous Bytes, and Verifying Storage Pages	6-21
6-9	Committing Changes, Exiting from RdbALTER, and Performing an RMU Verify operation of the EMP_INFO Storage Area of mf_personnel	6-21
7-1	Using Checksum to Verify Database Pages During a Backup Operation	7-32
7-2	Starting a Full Database Backup Operation	7-33
7-3	Starting an Incremental Backup Operation	7-36
7-4	Mounting a Tape Volume and Starting an Incremental Backup Operation	7-36
7-5	Determining the Date and Time of the Last Full Backup File	7-40
7-6	Displaying an Area to Determine a Timestamp	7-40
7-7	Starting a Parallel Backup Operation	7-48
7-8	Parallel Backup Plan File	7-50
7-9	Setting the Lock Timeout Interval to 300 Seconds	7-61
7-10	Starting an Offline Backup Operation When a User Is Attached to the Database	7-61

7-11	Performing a Full Backup Operation to One Tape	7-63
7-12	Performing a Full Backup Operation to Multiple Tapes Mounted on a Single Tape Drive	7-65
7-13	Performing a Full Backup Operation to Multiple Tapes Mounted on Multiple Tape Drives	7-66
7-14	Sample Oracle RMU Dump Backup_File Commands	7-77
7-15	Checking the Backup File to See When It Was Created	7-79
8-1	Starting a Full Restore Operation from the DBS_BACKUPS Disk	8-6
8-2	Omitting the New_Version Qualifier in a Restore Operation When a Previous Version of the Database Exists	8-9
8-3	Using the New_Version Qualifier During a Restore Operation to Supersede a Previous Version of the Database	8-9
8-4	Deleting an Old Database Version After Fully Restoring and Recovering the Database	8-9
8-5	Displaying the Root File Header to Check If the Database Was Ever Incrementally Restored	8-10
8-6	Starting an Incremental Restore Operation Following a Full Restore Operation	8-10
8-7	Displaying the Root File Header to Check if the Database Was Restored Incrementally	8-11
8-8	Error If You Try to Incrementally Restore the Same Database a Second Time, Using the Same Incremental .rbf File	8-12
8-9	Sample Restore Procedure Followed by Deleting an Old Version of the Database	8-13
8-10	Restoring and Recovering the EMP_INFO Storage Area	8-15
8-11	Error If You Try to Restore a Storage Area That Is Not Included in the Backup File	8-16
8-12	Error If You Try to Verify the Database	8-17
8-13	Restoring an Inconsistent Storage Area from a Backup File Containing That Storage Area	8-17
8-14	Verifying the Database and Checking the Version Number of the Storage Area File	8-17
8-15	Online By-Area Restore Operation of a Read/Write Storage Area	8-18
8-16	Performing a Restore-Only Root Operation and Initializing the TSN and CSN Values to Zero	8-28
8-17	Performing a Restore-Only Root Operation and Specifying .rda File, WORM, and .snp File Parameters	8-29
8-18	Using a Single Tape Drive for a Full Restore Operation	8-33
8-19	Using Multiple Tape Drives for a Full Restore Operation	8-34

8-20	Modifying After-Image Journaling Characteristics During a Restore Operation	8-40
8-21	Disabling After-Image Journaling During a Restore Operation	8-41
8-22	Changing the After-Image Journal File Specification During a Restore Operation	8-41
8-23	Restoring a List Storage Area to a WORM Optical Disk Device from a Read/Write Disk Device	8-44
8-24	Restoring a List Storage Area to a Read/Write Disk Device from a WORM Optical Disk Device	8-45
8-25	Using an Options File to Restore a Database	8-46
8-26	Using an Options File to Relocate All Database Files to Restore a Database	8-47
8-27	Making a Duplicate Copy of the Database and Moving Database Files to New Locations During a Restore Operation	8-48
8-28	Moving Database Files During a Restore Operation	8-49
8-29	Moving Dictionary Information During a Restore Operation	8-50
8-30	Checking That Metadata Was Moved into the Dictionary After a Restore Operation	8-50
8-31	Restoring a Database Without Updating the Data Dictionary	8-51
9-1	Enabling After-Image Journaling Using SQL Statements	9-6
9-2	Enabling After-Image Journaling Using RMU	9-7
9-3	Placement of the After-Image Journal File	9-10
9-4	Automated Custom Backup Procedure	9-33
9-5	After-Image Journal File Information Statistics Screen	9-41
9-6	Active User Stall Messages Screen	9-42
9-7	Restoring and Recovering a Lost Storage Area	9-61
9-8	Recovering a Database and Specifying the Device and File Specification of Journal File	9-67
9-9	Detecting Index Errors in an Inconsistent Database	9-68
9-10	Performing a Journal File Switch Increments the After-Image Journal Sequence Number	9-71
9-11	Displaying the AIJ Sequence Number	9-72
9-12	Displaying the Contents of an Empty Journal File	9-78
9-13	Displaying the Contents of a Journal File	9-79
9-14	Using Journal Files and Backup Files for a Full Database Backup Operation	9-82
9-15	Restoring, Verifying, and Recovering a Database	9-85

10-1	Changing the Number of Recovery Buffers Allocated to the Recovery Process	10-8
10-2	Displaying the Contents of an Empty Recovery-Unit Journal File . . .	10-9
10-3	Displaying the Contents of a Recovery-Unit Journal File	10-10
11-1	Common Display Format for Logical Areas, Snapshot Files, and Database Pages	11-2
11-2	Storage Areas of the mf_personnel Database	11-8
11-3	EMP_INFO Storage Area	11-10
11-4	CANDIDATES Logical Area	11-13
11-5	Commands to Display the First Page of All Logical Areas	11-15
11-6	Command to Display Area IDs of All Logical Areas	11-15
11-7	Logical Area RDB\$AIP Listing Logical Area Numbers and Names of the mf_personnel Database	11-16
11-8	Attempts to Display the Contents of the EMPLOYEES Logical Area	11-17
11-9	First Page of Selected Snapshot Files	11-18
11-10	Displaying Selected Pages of the EMPIDS_LOW Snapshot File to an Output File	11-22
11-11	Snapshot File Showing the Page Tail	11-23
12-1	Page Header for a Data Storage Page	12-3
12-2	Line Index for a Data Storage Page	12-4
12-3	TSN Index for a Data Storage Page	12-5
12-4	Locked and Unlocked Free Space for a Data Storage Page	12-7
12-5	User-Stored Data Storage Segment	12-9
12-6	Primary and Secondary Chained Segment	12-13
12-7	Primary Indexed Segment	12-14
12-8	Secondary Indexed Segment	12-16
12-9	Simple Data Segment	12-17
12-10	Data Segments Referenced by Primary and Secondary Indexed Segments	12-18
12-11	Sorted Index Node Segment for a Non-Ranked Sorted Index	12-21
12-12	Sorted Index Node Segment for a Ranked Sorted Index	12-24
12-13	Storage Area Page with Mixed Page Format Containing Hashed Index Node and Data Storage Records	12-26
12-14	Page Tail for a Data Storage Page in a Mixed Storage Area	12-35
12-15	Page Tail for a Data Storage Page in a Uniform Storage Area	12-36
12-16	Fragmented Storage Record	12-37
13-1	SPAM Page for a Storage Area with Uniform Page Format	13-6

13-2	ABM Page	13-9
13-3	An Area Inventory Page.....	13-10
13-4	SPAM Page for a Storage Area with Mixed Page Format	13-12
13-5	Storage Area SPAM Threshold Parameters for Specific Storage Areas	13-18
A-1	Using the SEARCH Command to Find the Exception in an Oracle Rdb Run-Time Services Bugcheck Dump File	A-3
A-2	Error Message for an Oracle Rdb Run-Time Services Bugcheck Dump	A-3
A-3	Error Message for an RMU Bugcheck Dump	A-4
A-4	Defining the RDMSBUGCHECK_DIR Logical Name	A-5
A-5	Exception Reports Extracted from RDMDBRBUG.DMP Files	A-6
A-6	Exception Generated from Exceeding the Disk Quota	A-7
A-7	Batch Command Procedure to Create Four Work Files, Each on a Separate Disk Volume	A-8
A-8	Exception Generated from Exceeding the Paging File Quota (PGFLQUOTA).....	A-9
A-9	Using an OpenVMS Text Editor to Read the Bugcheck Dump File.....	A-10

Figures

7-1	Backup and Restore Operations	7-1
7-2	Multithreaded Oracle RMU Backup (Single Process)	7-3
7-3	Multithreaded Oracle RMU Backup Using Multiple Parallel Processes	7-6
9-1	Restoring Data from the After-Image Journal Files	9-2
9-2	When to Back Up the After-Image Journal File	9-23
9-3	Checkpoint Processing and the Journal File Backup Procedure.....	9-56
11-1	Storage Area with Uniform Page Format	11-4
11-2	Mapping AIP and ABM Data Structures to SPAM and Data Pages	11-5
11-3	Storage Area with Mixed Page Format	11-6
12-1	Data Storage Page Components in a Mixed Page Format Storage Area	12-2
12-2	Adding a New Row to the Database	12-31
12-3	Oracle Rdb Page Structure with the System Record	12-33
12-4	Pointer to Another Page.....	12-34

13-1	SPAM Intervals in a Mixed Storage Area Using Defaults	13-15
13-2	SPAM Page Distribution with a Larger Interval	13-16

Tables

1-1	Database Administrator Responsibilities	1-1
1-2	Management Requirements	1-2
1-3	Database Management Tools	1-3
1-4	Database Maintenance Activities	1-4
3-1	Oracle RCU Functions That Are Not Audited	3-6
3-2	Daccess Privileges for Database Objects	3-10
3-3	Columns for Storing Security Audit Journal Records	3-22
5-1	Parameters for the Generated Command File	5-12
7-1	Types of Oracle RCU Backup Operations	7-9
7-2	Files, Areas, and Pages That Oracle RCU Backs Up	7-12
7-3	Comparison of Oracle RCU Backup and Operating System Backup Utilities	7-14
7-4	Determining the Frequency of Database Backups	7-16
7-5	Determining When to Use a Full or an Incremental Backup Operation	7-18
7-6	Strategies for Backup File Management	7-20
7-7	Comparison of Online and Offline Backup	7-23
7-8	Recommendations for Safeguarding Database Integrity	7-26
7-9	Sample Full and Complete, and Incremental Backup Procedures	7-27
7-10	Calculating Working Set Size	7-29
7-11	Recommended OpenVMS Process Quotas	7-30
7-12	Qualifiers for By-Area Backup Operations	7-43
7-13	Preparing for Parallel Backup Operations	7-45
7-14	Commands and Qualifiers for a Parallel Backup Operation	7-46
7-15	Starting an Online Backup Operation	7-57
7-16	Implicit and Explicit Tape Labeling	7-70
7-17	Oracle RCU Procedure to Check Tape Labels	7-74
9-1	Comparison of Single Versus Multiple After-Image Journal Files	9-11
10-1	Default Location for Recovery-Unit Journal Files	10-2
12-1	Components of a Data Storage Page	12-1
13-1	SPAM Entry for a Data Page	13-19

Send Us Your Comments

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

You can send comments to us in the following ways:

- **Electronic mail** — nedc_doc@us.oracle.com
- **FAX** — 603-897-3334 Attn: Oracle Rdb Documentation
- **Postal service**

Oracle Corporation
Oracle Rdb Documentation
One Oracle Drive
Nashua, NH 03062
USA

If you like, you can use the following questionnaire to give us feedback. (Edit the online release notes file, extract a copy of this questionnaire, and send it to us.)

Name _____ Title _____

Company _____ Department _____

Mailing Address _____ Telephone Number _____

Book Title _____ Version Number _____

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?

- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available).

Preface

Oracle Rdb7 Guide to Database Maintenance is one of a set of Oracle Rdb documents that describe database administrator responsibilities and tools for Oracle Rdb databases running on OpenVMS and Digital UNIX.

Purpose of This Manual

The purpose of this manual is to explain how to manage an Oracle Rdb database to perform operations such as:

- Monitoring database activity
- Performing security auditing
- Opening and closing a database
- Verifying and altering a database
- Backing up, restoring, and recovering a database
- Journaling database activity
- Displaying database pages
- Handling bugcheck dumps

Intended Audience

This manual addresses database administrators responsible for routine management with the goal of keeping Oracle Rdb databases available to run the company's business applications.

To use this manual most effectively, you need to be familiar with basic database management concepts and terminology, and Oracle RMU, the Oracle Rdb database management utility.

Operating System Information

Oracle Rdb is supported on OpenVMS VAX, OpenVMS Alpha, and Digital UNIX operating systems.

The *Oracle Rdb7 Installation and Configuration Guide* and *Oracle Rdb7 Release Notes* provide information about the versions of the supported operating systems and optional Oracle software compatible with this version of Oracle Rdb.

For information on the compatibility of other software products with this version of Oracle Rdb, contact your Oracle Rdb representative.

Structure

This manual contains 13 chapters, 1 appendix, and an index.

Chapter 1	Introduces database maintenance tasks.
Chapter 2	Describes how to monitor your database.
Chapter 3	Describes how to establish security auditing. This chapter also explains how to monitor and collect security audit records for many specific operations performed on a database, or how to send alarms to security operators' terminals, alerting operators to significant events as they happen.
Chapter 4	Describes how to open and close a database and the process of attaching to and detaching from a database.
Chapter 5	Describes how to perform regular verify operations and how to troubleshoot and solve problems detected by verify operations.
Chapter 6	Describes how to repair a database if it is corrupt.
Chapter 7	Describes how to perform backup operations.
Chapter 8	Describes how to restore your database.
Chapter 9	Describes after-image journaling and recovery techniques, including information about fixed-size journal files and journal switchover.
Chapter 10	Describes recovery-unit journaling and automatic recovery of .ruj files and explains how to improve recovery performance.
Chapter 11	Explains how to show the internal representation of a storage area and how to interpret what you see.
Chapter 12	Explains the internal representation of a storage page and how to interpret what you see.
Chapter 13	Explains the internal representation of a database SPAM page and how to interpret what you see.

Appendix A Describes how to handle an Oracle Rdb bugcheck dump.

Online Help

Online help can be invoked from the command line as well as through interactive Oracle Rdb utilities. Refer to the *Oracle Rdb7 Release Notes* for information about accessing online help.

Related Manuals

This manual, together with the following manuals, describe all facets of Oracle Rdb database administration:

- *Oracle RMU Reference Manual*
- *Oracle Rdb7 SQL Reference Manual*
- *Oracle Rdb7 Guide to Database Design and Definition*
- *Oracle Rdb7 Guide to Database Performance and Tuning*

Also, refer to the *Oracle Rdb7 Release Notes* for release notes and other information specific to this release of Oracle Rdb.

For businesses that require uninterrupted (24x365) database services, you should refer to *Oracle Rdb7 and Oracle CODASYL DBMS: Guide to Hot Standby Databases*. This manual describes the optional Oracle Rdb capability to automatically duplicate your master Oracle Rdb database with a redundant database located at a geographically remote site.

Conventions

In this manual, Oracle Rdb refers to Oracle Rdb for OpenVMS and Oracle Rdb for Digital UNIX software. Version 7.0 of Oracle Rdb software is often referred to as V7.0.

The SQL interface to Oracle Rdb is referred to as SQL. This interface is the Oracle Rdb implementation of the SQL standard ANSI X3.135-1992, ISO 9075:1992, commonly referred to as the ANSI/ISO SQL standard or SQL92.

Oracle CDD/Repository software is referred to as the dictionary, the data dictionary, or the repository.

OpenVMS means both the OpenVMS Alpha and OpenVMS VAX operating system.

This manual uses icons to identify information that is specific to an operating system or platform. Where material pertains to more than one platform or operating system, combination icons or generic icons are used. For example:



This icon denotes the beginning of information specific to the Digital UNIX operating system.



This icon combination denotes the beginning of information specific to both the OpenVMS VAX and OpenVMS Alpha operating systems.



The diamond symbol denotes the end of a section of information specific to an operating system or platform.

Discussions in this manual that refer to VMScluster environments apply to both VAXcluster systems that include only VAX nodes and VMScluster systems that include at least one Alpha node, unless indicated otherwise.

In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the Return key at the end of a line of input.

Often in examples the prompts are not shown. Generally, they are shown where it is important to depict an interactive sequence exactly; otherwise, they are omitted in order to focus full attention on the statements or commands themselves.

The following conventions are also used in this manual:

- . Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
- ...
- Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted.
- boldface text** Boldface type in text indicates a term defined in the text, the glossary, or in both locations.

< >	Angle brackets enclose user-supplied names.
[]	Brackets enclose optional clauses from which you can choose one or none.
\$	The dollar sign represents the DIGITAL Command Language prompt on OpenVMS systems and the Bourne shell prompt on Digital UNIX systems.
#	The number sign represents the Digital UNIX default superuser prompt, and is also used as a comment character in the Bourne shell.
%	The percent sign represents the Digital UNIX default user prompt.
>	In examples, when the right angle bracket appears at the beginning of a continued line, it represents the Bourne shell command line continuation prompt. Otherwise, it represents a redirection operation.
UPPERCASE	The Digital UNIX operating system differentiates between lowercase and uppercase characters. Examples, syntax descriptions, function definitions, and literal strings that appear in text must be typed exactly as shown.
lowercase	
filesystem	In text, this typeface indicates the exact name of a command, routine, partition, path name, directory, or file. This typeface is also used in interactive examples and other screen displays.

Technical Changes and New Features

This section lists some of the new and updated portions of this manual since it was last released with Version 6.0 of Oracle Rdb.

The following changes have been made to this manual:

- The overview information in Chapter 1 has been rewritten to focus on the availability features of Oracle Rdb.
- Chapter 7 about database backup operations has been rewritten to provide an overview of the default backup operation and the new parallel backup operation. The chapter also offers suggestions for backup strategies, a sample backup procedure, tape checking procedures, and calculations for determining the appropriate working set size for OpenVMS system backup operations. Also, Chapter 7 contains new information about cluster and multiple-volume backup operations.
- Chapter 9 about after-image journaling has been rewritten to provide a more comprehensive look at after-image journaling as a part of your total restore and recovery procedure. The chapter contains new information about the use of multiple fixed-size journal files and includes information about how Oracle Rdb automatically switches to a new journal file when the current journal file becomes full. The chapter describes how you can set up an *emergency* after-image journal file to avoid journaling suspension.
- Chapter 10 about recovery-unit journaling has been rewritten to provide information about how Oracle Rdb creates the .ruj file and the default location.
- The chapter titled *Understanding Internal Database File Structures* has been split into three chapters:
 - Chapter 11 explains the internal representation of database storage areas.
 - Chapter 12 explains the internal representation of database storage pages.

- Chapter 13 explains the internal representation of database SPAM pages.
- Section 13.3 provides new calculations for determining the actual number of pages in the SPAM interval.
- Chapter 7 describes the following qualifiers that are new or changed for the RMU Backup command:
 - RMU Backup Parallel
 - RMU Backup Plan
 - RMU Backup Loader_Synchronization
 - RMU Backup Media_Loader
 - RMU Backup Quiet_Point
- Section 7.8.2 describes how the Noscan_Optimization and Scan_Optimization qualifiers are ignored during online, full backup operations. Oracle RMU returns an information message indicating that the [No]Scan_Optimization qualifier is ignored when you also include the Online qualifier for a full backup operation. The current recording state at the time you enter the Backup command remains in effect.
- Section 8.2 shows you how to provide sufficient Oracle RMU privileges to a restored database. Instructions and examples show how to provide privileges to a restored database owned by a resource identifier.
- Chapter 11 and Chapter 12 provide information about how the uniform page format contains many records from only one table or one index. The exception is for indexes created in the RDB\$SYSTEM area. In this area, a uniform format page may contain records (for example, nodes) from multiple indexes if they are defined for the same table.
- Chapter 12 explains new on-disk structure for B-tree indexes.

New for Version 7.0, you can specify that Oracle Rdb use a new B-tree on-disk structure. The new structure allows better optimization of queries, particularly queries involving range retrievals. Oracle Rdb is able to make better estimates of cardinality, reducing disk I/O and lock contention.

Chapter 12 explains ranked and non-ranked sorted index node structures and the output from the RMU Dump command that displays sorted index node segments.
- Section 10.1 discusses ways to improve the performance of the automatic recovery processes and the new default WSEXTENT quota for the DBR process of 8192 pages.

If a node failure causes Oracle Rdb to initiate the DBR process on a node other than the one where the process is failing, the DBR process cannot access the working set extent (WSEXTENT) quota from the failing process. This is because the process (on the failing node) no longer exists. Because the DBR process is unable to inherit the failing process' WSEXTENT value, it defaults to 8192 pages.

Refer also to the *Oracle Rdb7 Release Notes* for additional information about new features, technical changes, current limitations, and restrictions.

Oracle Rdb Database Maintenance

The need for efficient database management has become more urgent as systems grow in physical size, capacity, and complexity. Most businesses store critical data in database management systems. These databases often are shared among multiple office locations and used by large numbers of people performing varied tasks. Consequently, the businesses require that the database is functioning properly and is available for use at all times.

This chapter summarizes database management techniques and tools that help the database administrator (DBA) properly manage all the requirements of an Oracle Rdb database.

1.1 Database Administrator Responsibilities

Database administration involves planning, designing, implementing, maintaining, and tuning one or more databases.

Table 1–1 lists broad categories of DBA responsibilities and indicates the Oracle Rdb manual that provides information about the topic.

Table 1–1 Database Administrator Responsibilities

DBA Task	Involves...	Reference
Design or reorganize the database	Initiating a complete analysis of the business needs and data requirements in order to achieve an appropriate database design and definition. Reorganizing or changing mature databases to reflect changing business needs.	<i>Oracle Rdb7 Guide to Database Design and Definition</i>
Implement the database	Fine tuning the definition of the database and its entities, setting up security mechanisms, setting up a data repository, and loading data.	<i>Oracle Rdb7 Guide to Database Design and Definition</i>

(continued on next page)

Table 1–1 (Cont.) Database Administrator Responsibilities

DBA Task	Involves...	Reference
Maintain the database	Opening and closing a database, data backup and restoration, security auditing, and journaling; tasks that ensure the database availability, integrity, security, and scalability required to meet your business needs.	<i>Oracle Rdb7 Guide to Database Maintenance</i>
Tune the database	Collecting data usage statistics, testing performance, improving database access, and improving applications to attain optimal performance.	<i>Oracle Rdb7 Guide to Database Performance and Tuning</i>
Distribute or move the database	Distributing applications across a network distributed environment, or moving a database from a development environment to a production environment.	<i>Oracle Rdb7 Guide to Distributed Transactions and Oracle Rdb7 Guide to Database Performance and Tuning</i>
Provide a fault-tolerant database	Physically replicating a database, applications, and environment at a geographically remote standby site.	<i>Oracle Rdb7 and Oracle CODASYL DBMS: Guide to Hot Standby Databases</i>

This manual concentrates on the database administrators job regarding the ongoing management of the database.

1.2 Database Management Requirements

Your overall goal as a DBA is to maintain and protect the integrity of the data and the availability of the database. Database management requirements are shown in Table 1–2.

Table 1–2 Management Requirements

Requirement	Description
Analysis	Collect and display information about database users, data, and characteristics.
Recovery	Return the database to a correct, consistent state after a hardware or software failure, or human error.
Integrity	Ensure that data is correct. The database must not store data that is inconsistent with the rules of the business. Once stored in the database, data must persist. Partially entered transactions are rolled back.
Security	Protect data from inadvertent or deliberate destruction or viewing by an unauthorized person.

(continued on next page)

Table 1–2 (Cont.) Management Requirements

Requirement	Description
Concurrency	Ensure that multiple users and transactions can access the database concurrently and see a consistent view of the data.

1.3 Database Maintenance Tools

Once a database is installed and in use, it must be monitored and maintained. You maintain an Oracle Rdb database using a combination of data manipulation languages (for example, SQL) and Oracle RMU, the Oracle Rdb database management utility.

Table 1–3 lists general categories of database maintenance tasks, and the Oracle Rdb utilities and tools that help you perform the maintenance operations.

Table 1–3 Database Management Tools

Operation	Utility	Purpose
Analyze	RMU Analyze	Collects and displays information about how data is being stored and about the index structure
	RMU Show	Displays information about the Oracle Rdb database running on the node from which you issue the Show command
	RMU Verify	Checks internal database structures for consistency and integrity
	RMU Dump	Displays information about the contents, structure, and users of the database
Control	RMU Open	Manually opens the database for normal user access
	RMU Close	Manually closes the database for maintenance and restricted access
	RMU Monitor	Starts or stops the Oracle Rdb monitor process
	RMU Set	Allows you to control AIJ login, audit login, corrupt pages, and privileges
Integrity	RMU Backup	Creates full or partial (incremental) backup copy of the database or after-image journal file

(continued on next page)

Table 1–3 (Cont.) Database Management Tools

Operation	Utility	Purpose
	RMU Restore	Restores the database to its state at the beginning of the execution of RMU Backup
	RMU Optimize	Recovers the database to the last committed transaction
	RMU Resolve	Eliminates unneeded and duplicate journal records and orders the journal records
	RMU Server After_Journal	Commits or rolls back any unresolved distributed transactions in the database, and manually starts or stops the AIJ log server (ALS) process for the specified database
Load	RMU Load	Loads data from a file into an Oracle Rdb table
Unload	RMU Unload	Unloads data from a specific table or view into a file
Restructure or Update	SQL Import and Export	Assists with major restructuring or migration of a database
	RMU Convert	Upgrades an Oracle Rdb database from a previous version to the current version
	RMU Extract	Reads and decodes Oracle Rdb metadata and reconstructs equivalent statements in RDO or SQL
Replication	RMU Copy_Database	Manually replicates a database
Replication	RMU Replicate	Automatically replicates a database

1.4 Database Maintenance Tasks

Table 1–4 lists database maintenance tasks according to when they should be performed (daily, weekly, monthly, and so forth). Table 1–4 also shows the Oracle Rdb utilities and tools that help you perform the maintenance operations.

Table 1–4 Database Maintenance Activities

Daily Maintenance	Method
Perform incremental backup operation of database	RMU Backup/Incremental
Attach to database, map root file, map OpenVMS global sections or Digital UNIX shared memory partitions	RMU Open

(continued on next page)

Table 1–4 (Cont.) Database Maintenance Activities

Daily Maintenance	Method
Detach from database, unmap OpenVMS global sections or Digital UNIX shared memory partitions, abort user processes	RMU Close
Report on database activity	RMU Show or on OpenVMS systems you can type SYSSYSTEM:RDMMON411.LOG
Perform incremental verification of database integrity	RMU Verify Incremental
Weekly Maintenance	Method
Perform full backup operation of database	RMU Backup
Report on database space usage	RMU Analyze
Display performance indicators interactively	RMU Show Statistics
Perform full verification of database integrity prior to an RMU Backup operation	RMU Verify All
Database Startup	Method
Create monitor process, start monitor log	RMU Monitor Start (Use at system startup)
Attach to database, map root file, map OpenVMS global sections or Digital UNIX shared memory partitions	RMU Open (Use after restart)
Database Shutdown	Method
Detach from database, unmap OpenVMS global sections or Digital UNIX shared memory partitions, abort user processes	RMU Close (Use before maintenance operations)
Terminate the monitor process, let users currently attached finish, and prevent new users from attaching to the database	RMU Monitor Stop (Use at system shutdown)

(continued on next page)

Table 1–4 (Cont.) Database Maintenance Activities

Troubleshooting	Method
Restore database	RMU Restore (Use after unrecoverable loss of database)
Restore root file	RMU Restore Only_Root (Use after unrecoverable loss of root file)
Roll database forward	RMU Recover (Use after RMU Restore operation)
Report on database integrity after a system failure or on database integrity of a restored database after the RMU Restore operation	RMU Verify (Use after system failure or after using the RMU Restore operation)
Patch the database if the corruption is minor in scope	RMU Alter (Use after verification reports a corruption problem)
Report on database space usage	RMU Analyze (Use after noticing performance problems)
Display performance indicators interactively	RMU Show Statistics (Use after noticing performance problems)
Display contents of database, storage area, and .SNP files, including root information	RMU Dump (Use after noticing performance problems)
Rebuild SPAM and ABM pages, fix page tail problems, create and initialize new snapshot (.snp) files for those that are missing	RMU Repair (Use after noticing SPAM page corruption, area bit map (ABM) page or page tail errors, or missing .snp files)
Set pages corrupt; set pages consistent	RMU Set Corrupt_Pages (Use after verification reports a corruption problem)
Display corrupt pages by disk, area, and page	RMU Show Corrupt_Pages (Use after verification reports a corruption problem, corrupt pages have been previously set, or a display of the header file indicates corrupt pages exist)

(continued on next page)

Table 1–4 (Cont.) Database Maintenance Activities

General RMU Maintenance	Method
Create a new version of the monitor log file	RMU Monitor Reopen_Log
Display contents of database files	RMU Dump
Report on current database users	RMU Show Users
Report on users of all databases	RMU Show System
Copy a table or view into a BRP format or RMS file	RMU Unload
Copy data unloaded using the RMU Unload command into a table	RMU Load
Convert an existing database file to a format compatible with a new version of Oracle Rdb	RMU Convert
Create a backup copy of the database prior to conversion and a backup copy of the database immediately following a successful conversion and verification	RMU Backup
Create a backup file of the database .aij file	RMU Backup After_Journal
Improve the performance of rolling forward .aij files	RMU Optimize After_Journal
Update the cardinality in the metadata when the storage area has been set to read-only and when cardinality values stored in the system tables no longer accurately reflect the characteristic of the data stored in the database	RMU Collect Optimizer Statistics
Copy a database on line	RMU Copy_Database Online
Move storage areas of a database when the database is off line	RMU Move_Area
Enable Oracle Rdb security auditing	RMU Set Audit
Display the set of Oracle Rdb security auditing characteristics established with the RMU Set Audit command	RMU Show Audit
Enable Oracle Rdb to load security audit records from the security audit journal into a table in your database	RMU Load Audit
Enable modification of the root file access control list (ACL) for a database	RMU Set Privilege

(continued on next page)

Table 1–4 (Cont.) Database Maintenance Activities

General RMU Maintenance	Method
Decode system table information and reconstruct equivalent commands in the selected interface (SQL or RDO) for the definition of that database	RMU Extract
Force all active database processes on all nodes to immediately perform a checkpoint operation	RMU Checkpoint
Enable setting of AIJ attributes	RMU Set After_Journal
Display AIJ configuration information and optionally initialize AIJ symbols	RMU Show After_Journal

General SQL Maintenance	Method
Modify schema and database characteristics	SQL ALTER DATABASE
Copy a database into an intermediate, compressed interchange (.RBR) file format for migration or restructuring	SQL EXPORT DATABASE
Copy the contents of the .RBR files created with the EXPORT statement into a database to complete database migration and restructuring	SQL IMPORT DATABASE
Copy a database into an intermediate, compressed .RBR file format for migration or restructuring, but contain only metadata and no data	SQL EXPORT DATABASE NO DATA
Copy the contents of the .RBR files (with data) created with the EXPORT statement into a database to complete database migration and restructuring, but contain only metadata and no data	SQL IMPORT DATABASE NO DATA
Update database access control by adding new database users or modifying user privileges	SQL GRANT
Update database access control by deleting database users or modifying user privileges	SQL REVOKE
Reorganize the relation records or table rows within one or more storage areas according to partitions specified	SQL ALTER STORAGE MAP REORGANIZE

1.5 Database Availability

Availability is the amount of time that a computing system provides application service to its users. Making the database continuously available to users and applications is a primary goal of database administration.

The availability of a database management system can be compromised not only by unscheduled downtime such as hardware or software failures, but also by routine maintenance tasks such as backup operations, definition changes, and file or data restructuring.

Some database management systems are restricted by their degree of fault tolerance. Oracle Rdb has few restrictions and provides a high degree of availability. The following sections describe the availability features of Oracle Rdb and how they minimize the impact of scheduled and unscheduled downtime on database access.

1.5.1 Fault-Tolerant Oracle Rdb Databases

Oracle Rdb provides fault tolerance by allowing you to physically duplicate a database, applications, and environment at a geographically remote standby site to provide fault tolerance. In the event of a failure, Rdb continues to provide the required services by transparently failing over users and applications to the replicated database.

You use Oracle RMU to replicate a database at a remote site and automate after-image journal backup and rollforward operations to provide a nonintrusive, high-performance solution to data availability. See the *Oracle Rdb7 and Oracle CODASYL DBMS: Guide to Hot Standby Databases* manual for complete information.

1.5.2 Online Backup Operations

Oracle Rdb supports online backup operations that allow users access to the database during that operation. The online backup operation uses snapshot (.snp) files to achieve a high degree of database availability. Oracle Rdb also supports fast online or offline incremental backup operations, which reduce backup time and ease maintenance by reducing the amount of data that needs to be backed up. Fast incremental backup operations use memory bitmaps for each storage area's space area management (SPAM) pages in the global section or shared memory partition of each node.

Oracle Rdb can provide the fastest and most reliable backup capabilities of any database system on Alpha and VAX computers. Oracle Rdb can back up multiple storage areas at the same time, concurrently running several tape drives through its support of the multithreaded backup operation.

Note

The number of tape drives that can be used concurrently is limited only by the system's I/O capacity.

Refer to Chapter 7 for more information about backing up your database.

1.5.3 Online By-Area and By-Page Restore and Recovery Operations, and By-Area Move Operations

Oracle Rdb supports online by-area and by-page restore and recovery operations as well as online move storage area operations. If a storage area or pages within a storage area need to be restored and recovered, or if a storage area needs to be moved for some reason, the database need not be shut down and made inaccessible to users during these maintenance operations. In fact, only pages within an area or areas that are being restored and recovered become inaccessible until the restore and recover operation completes. Similarly, only areas that are moved are inaccessible until the move operation completes. Once the area or pages within an area are successfully restored and recovered, or the area is moved, it is accessible to users again. See Chapter 8, Chapter 9, and the *Oracle Rdb7 Guide to Database Design and Definition* for more information.

1.5.4 Disabling Journaling for Write-Once Storage Areas

Multimedia applications can store very large amounts of list data (images, documents, video, voice, and so forth). If after-image journaling is enabled, sufficient storage space must be available to handle the large .aij files that can result. Also, additional tape media may be required to back up these large .aij files. For some applications, this may be impractical. An alternative is to disable journaling of list data stored in write-once storage areas on write-once, read-many (WORM) devices while continuing to journal all other database activity.

1.5.5 Cluster and Networkwide Automatic Recovery

Oracle Rdb has supported fault tolerance in a cluster environment since V2.0 (1985) of Oracle Rdb, so if a CPU goes off line or an HSC node fails, Oracle Rdb automatically handles recovery and rollback. Refer to the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information on this process.

1.5.6 Automatic Cleanup

The following online activities are automatically performed by the database while users are attached to it:

- Use of freed space in database storage area (.rda) files and snapshot (.snp) files
- Creation, extension, or deletion of a user's recovery-unit journal (.ruj) file, as required
- Extension of any .rda files, as required

- Extension of an .aij file, as required
- Recovery of any aborted transactions
- Change in lock granularity of a user's transaction, as required
- Detection of deadlocks
- Updates to approximate cardinality of tables and indexes, as appropriate
- Group commit operations and after-image journal data for efficiency

See Chapter 9, Chapter 10, the *Oracle Rdb7 Guide to Database Design and Definition*, and the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information.

1.5.7 Online DBA Activities

As a DBA, you can perform the following activities on line while users are attached to the database:

- Perform online backup operations of the database, either full or incremental backup operations.
- Perform online restore operations of storage areas.
- Perform online recovery operations of restored storage areas.
- Perform online spooling of the .aij file (that is, truncate and back up contents of the .aij file).
- Set the following security auditing characteristics:
 - Enable or disable security auditing.
 - Enable or disable security alarms.
 - Check each user's first access for the specified audit objects.
 - Check for forced write operations of audit journal records.
 - Set security audit event class flags.
 - Set audit security alarm names.
 - Set the security audit file name.
- Create an online copy of a database.
- Monitor and record performance statistics.
- Display any database area (header, indexes, and data areas).
- Change the lock timeout interval (a deferred action seen only by new users).

- Change the following database characteristics:
 - Change the open mode of the database to either automatic or manual.
 - Change the default number of database buffers per user (a deferred action seen only by new users).
 - Change the default number of database recovery buffers (a deferred action seen only by new users).
 - Change allocation and extension characteristics for the .aij file.
 - Change the allocation characteristics for an .snp file.
 - Change the space allocation characteristics, such as extension values of the database.

Note

The database can dynamically extend as required, but the DBA can set a default value by which to extend the database.

- Change the global buffer count (a deferred action seen only by new users).
- Change the maximum number of global buffers per user (a deferred action seen only by new users).
- Add a journal (if multiple .aij files of a fixed size are being used and if there are available slots).
- Alter or drop a journal (cannot drop an active journal).
- Change journal backup server characteristic.
- Change journal backup file name.
- Change journal cache file name.
- Change journal extent.
- Change journal log server.
- Change journal notify procedure.
- Change journal overwrite procedure.
- Change the following storage area characteristics:
 - Enable or disable volume spreading.

- Enable or disable extension, and set extension parameters (minimum, maximum, and percent).
- Enable or disable journaling for write-once storage areas.
- Change the following system storage area (such as the RDB\$SYSTEM storage area) characteristics, if you have exclusive access to the storage area:
 - Set the read-only and write-once attributes (except storage area).
 - Change SPAM thresholds.
 - Truncate an .snp file (truncating an .snp file blocks read-only transactions).
 - Add or drop a storage area.
 - Enable or disable SPAMs.
- Check the database integrity by using the RMU Verify command (this operation must wait for exclusive and batch-update transactions to complete).
- Update metadata, as follows:
 - Create or drop catalogs.
 - Create, alter, or drop collating sequences.
 - Create or drop constraints.
 - Create, alter, or drop domains.
 - Create or drop functions.
 - Create, alter, or drop indexes.
 - Create or drop modules.
 - Create or drop query outlines.
 - Grant or revoke protection.
 - Create, alter, or drop tables.
 - Create or drop triggers.
 - Create, alter (certain parameters as previously specified), or drop storage areas, all with only exclusive access to the storage area.
 - Create, alter, or drop storage maps.
 - Create or drop views.

- Move a storage area.

See Chapter 3, Chapter 5, Chapter 7, Chapter 8, Chapter 9, the *Oracle Rdb7 Guide to Database Design and Definition*, and the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information.

1.5.8 Offline DBA Activities

As DBA, you must perform the following activities off line because they require the database to be shut down:

- Create, alter, or drop a database.
You can alter any of the database objects and database characteristics as stated in Section 1.5.7 with the database on line and while users are attached to the database.
- Change the following database characteristics:
 - Change the maximum number of database users.
 - Change the maximum number of cluster nodes that can access the database.
 - Change the database read-only or read/write attribute (storage area).
 - Enable or disable snapshots.
 - Change the snapshots deferred characteristic.
 - Change the global buffer enabled characteristic.
 - Change the lock carryover optimization characteristic.
 - Change checkpoint parameters.
 - Change fast commit parameters.
 - Change the characteristics of database locks (whether adjustable lock granularity is enabled or disabled).
 - Change the after-image journaling (drop and add the file name used or change the enable/disable option) for a single extensible .aij file.
 - Reserve journal files.
 - Reserve storage areas.
 - Change the statistics enabled or disabled characteristic.
 - Change page-level or row-level locking.

- Change the requirement for a dictionary to be used when metadata is changed.

Refer to the *Oracle Rdb7 Guide to Database Design and Definition* for more information on changing database characteristics, adding new storage areas, changing and deleting storage areas, adding and deleting indexes, and adding and deleting storage maps.

1.5.9 DBA Activities Requiring a Database Reload Operation

The following activities require the database to be shut down, unloaded with the SQL EXPORT statement, and reloaded with the SQL IMPORT statement:

- Changing the buffer size
- Changing the number of users on a single-file database (unload and reload operations are not required for a multifile database)
- Changing the number of cluster nodes for a single-file database (unload and reload operations are not required for a multifile database)

Refer to the *Oracle Rdb7 Guide to Database Design and Definition* for more information on using the SQL EXPORT and IMPORT statements.

1.5.10 Quick and Automatic Database Recovery

Oracle Rdb automatically detects abnormal termination of users' transactions and initiates rollback and recovery without causing data inconsistency. Refer to Chapter 10 for more information on journaling and recovery for update transactions, and the recovery-unit journal (.ruj) file.

In a cluster environment, Oracle Rdb will coordinate recovery even if the node that started the original transaction is no longer present, thereby ensuring recovery capability and database integrity in a cluster environment. In addition, database recovery processes (DBRs) start in parallel to recover from node failures. Refer to the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information on the automatic recovery procedure.

1.5.11 Database Integrity

Database integrity has been one of the primary features of Oracle Rdb since its inception. Automatic detection, rollback, and recovery of incomplete transactions through user-based recovery-unit journaling, after-image journaling, and use of the OpenVMS Distributed Lock Manager means that Oracle Rdb maintains optimum integrity in all OpenVMS environments. These environments include single or multiple CPU nodes, CI clusters, local area clusters, mixed clusters, and DECnet distributed environments. Each database

page contains a checksum, which is checked by Oracle Rdb and recalculated after each update.

Refer to Chapter 7 for a description and examples of database backup and verification, Chapter 9 for information on journaling and recovery, and the *Oracle Rdb7 Guide to Database Performance and Tuning* for information on using Oracle Rdb in a cluster environment.

1.5.12 Checking Database Integrity and Evaluating Performance

The Oracle RMU management utility includes the RMU Verify, RMU Dump, and RMU Analyze commands for checking database integrity and performing database analysis.

A database should be verified regularly. If problems are detected, the cause must be determined before the database is made accessible to users again. Verifying a database is discussed in Chapter 5. One method of isolating database integrity problems is to display the contents of the page that is corrupt for a more detailed inspection. Displaying database pages and interpreting the contents are described in Chapter 11 and Chapter 12. Information on database integrity is described in Chapters 7, 8, 9, and in the *Oracle Rdb7 Guide to Database Performance and Tuning*.

The performance of a database application should be monitored regularly to provide a benchmark for the application and to detect early signs of problems as the database grows or changes in size. The RMU Show Statistics command is used to analyze database performance. For more information about DBA tools for evaluating performance and database behavior, refer to the *Oracle Rdb7 Guide to Database Performance and Tuning*.

Occasionally, transactions may compete for resources and cause deadlocks to occur. Deadlocks are detected and broken by Oracle Rdb (through its use of the OpenVMS Distributed Lock Manager). Information on deadlocks and other lock statistics can be viewed and recorded using the RMU Show Statistics command. Refer to the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information on using this command and interpreting its results.

1.6 Creating Sample Single-File and Multifile Databases

This manual describes maintenance topics and provides examples from the sample personnel databases, single-file personnel and multifile mf_personnel, which are used throughout Oracle Rdb documentation. You can follow along and type in most of the examples if you have access to copies of these databases.

The sample personnel database created by the Installation Verification Procedure (IVP) is small and is a single-file database. It has only 9 tables, each containing from a few to 100 rows, and 3 views. For more information about the location and creation of the sample personnel databases, refer to the *Migrating Oracle Rdb7 Databases and Applications to Digital UNIX* or the *Migrating Oracle Rdb7 Databases and Applications to OpenVMS Alpha*.

Monitoring Your Oracle Rdb Databases

After your database is created and in use, programs and interactive SQL can access it using data manipulation language (DML) statements. The Oracle Rdb monitor process collects and maintains information about database use and overall activity.

This chapter describes the Oracle Rdb monitor process, lists the different types of monitor information you can display, tells you how to produce each type of monitor information, and explains how to use the RMU Show commands to display information about:

- Users of a specific database
- Users of all databases

The RMU Show commands and monitor log file display information for a single Alpha or VAX computer; they do not display information about an entire cluster environment. To obtain summary information about database activity over time, display the monitor log file using operating system commands (for example, the TYPE command on OpenVMS or the more command on Digital UNIX).

The RMU Dump commands display the following clusterwide information for any Oracle Rdb database:

- Users of a database throughout a cluster environment
- Current settings of database characteristics such as after-image journaling and lock options
- Contents of database files

2.1 The Oracle Rdb Monitor Process

The Oracle Rdb monitor runs under the SYSTEM account in a detached process called RDMS_MONITOR. The monitor process controls database access, initiates the automatic recovery procedure, and maintains a log of database activity. A monitor process must be running on your computer for you to use Oracle Rdb.

You start and stop the monitor process interactively using the RMU Monitor command.

OpenVMS

On OpenVMS systems, you execute the RMONSTART.COM command file that starts the monitor process from within your system's startup file, which is executed each time you start up your system.

For example, you execute the RMONSTOP.COM command file to stop the monitor from within your SYSSMANAGER:SYSHUTDWN.COM command file, which is executed each time you shut down your system. See the *Oracle Rdb7 Installation and Configuration Guide* for further information. ♦

Digital UNIX

On Digital UNIX systems, the monitor runs under the ownership of the dbsmgr account. Refer to the RMU Monitor command in the *Oracle RMU Reference Manual* for additional information. ♦

There is only one monitor process per version of Oracle Rdb installed per node, no matter how many databases there are.

If you want to shut down a single database and immediately recover recovery-unit journal (.ruj) files, use the RMU Close command with the Wait qualifiers. (On OpenVMS systems, you can shut down a single database clusterwide by also including the Cluster qualifier on the the RMU Close command.)

Use either the Abort=Delprc qualifier, to close the database immediately, or the Abort=Forcex qualifier, to close the database and allow process rundown. Then, use the RMU Monitor Stop command to shut down the database monitor. If you want to shut down all database activity on the computer without first closing the database, but you want to allow current user processes to continue and run down before stopping, use the RMU Monitor Stop command with the Noabort qualifier (see Section 2.1.1).

The monitor process keeps track of current database users. You can enter the RMU Show commands to access this information and display it. The monitor process records certain database activity, including the nature of each attach request, in the monitor log file.

If a database needs recovery—the monitor process determines this by reading the database root (.rdb) file—the attach request is denied or delayed until the recovery can be completed. The monitor also opens the database when it receives an attach request for a database that is not already open, provided the database administrator (DBA) has enabled automatic opening of the database.

2.1.1 Starting and Stopping the Monitor Process Interactively

You can start and stop the monitor process interactively using the RMU Monitor command.

OpenVMS

Note

Oracle Rdb recommends that you start the monitor from the SYSTEM account, which has SETPRV privilege. If you do not start the monitor from a SYSTEM account, you will need the following privileges: ALTPRI, CMKRNL, DETACH, PSWAPM, SYSGBL, SYSNAM, SYSPRV, and WORLD. If your nonsystem account lacks any of these privileges, the monitor process may not start, or it may start but not operate properly.

◆ Use the RMU Monitor Stop command to stop the monitor process interactively. Example 2–1 shows this command on an OpenVMS system.

Example 2–1 Stopping the Monitor Process

```
$ RMU/MONITOR STOP
```

When you stop the monitor interactively, by default no new Oracle Rdb processes are allowed to start, and normal process rundown is allowed for active users. No new users (including those issuing RMU commands) are allowed to access the database until the monitor is restarted.

Use the Wait qualifier to indicate that Oracle Rdb should wait until the local monitor shutdown is complete. The default is Nowait.

Use the Abort qualifier to abort active database user processes when you stop the monitor. The default is Noabort.

Example 2–2 shows how to use this command on a Digital UNIX system.

Example 2–2 Stopping the Monitor Process and Aborting User Processes

```
$ rmu -monitor stop -abort
```

On OpenVMS systems, you can abort user processes by forcing an image exit in one of the following two ways:

- **FORCEX**
Causes an exit handler to recover user processes that have been forced off the database system (from a forced process exit) and returns them to the operating system prompt (\$). FORCEX is the default.
- **DELPRC**
Deletes a user process and causes a recovery process to start the next time someone attaches to the database. DELPRC logs the user off the system.

For detailed information about the FORCEX and DELPRC system services, refer to the OpenVMS system services routines documentation. ♦

Any transactions that were active when the user processes were aborted are recovered the next time any Oracle Rdb user attaches to the database.

Use the RMU Monitor Start command to start the monitor process again interactively. Enter the command (the command for an OpenVMS system is shown in Example 2–3) from each node on which users access Oracle Rdb databases.

Example 2–3 Starting the Monitor Process

```
$ RMU/MONITOR START
```

2.1.2 Renaming the Monitor Log File

OpenVMS

Each time the monitor process starts, it creates a new version of the monitor log file. By default, the monitor log file name is SYSSSYSTEM:RDMMON.LOG in a standard or single-version environment. When Oracle Rdb is running in a multiversion environment, a 2-digit number is appended to the file name to represent the version number. For example, the monitor log file name in a multiversion environment for V7.0 is SYSSSYSTEM:RDMMON70.LOG. The SYSSSYSTEM logical name points to the SYSSSYSROOT:[SYSEXE] directory by default. The logical name SYSSSYSROOT points to a specific disk device and directory, such as \$111\$DUA1:[SYS1.], which equates to the SYSSCOMMON logical name. Therefore, to find the default location of the RDMMON.LOG file, set your default directory to SYSSSYSTEM.

You can display the contents of the current monitor log file by using the operating system TYPE command or a text editor, as shown in Example 2–4 (you need the OpenVMS SYSPRV privilege to display the monitor log file).

Example 2–4 Displaying the Contents of the Monitor Log File

```
$ TYPE SYSSSYSTEM:RDMMON.LOG
$
$ ! Use TPU to examine a running log file.
$
$ EDIT/TPU/READ_ONLY SYSSSYSTEM:RDMMON.LOG
```

You can specify a device and directory other than the default to receive the monitor log file by stopping and restarting the monitor interactively. If you do specify a different device and directory, as shown in Example 2–5, you can define a system-level logical name to point to the new location (you must have SYSNAM privilege). For example, define the system-level logical name as RDBMONITOR. You can also rename the log file itself. You may want to specify a different device and directory for the monitor log file if you have the disk space available.

Example 2-5 Specifying a Different Device and Directory for the Monitor Log File

```
$ RMU/MONITOR STOP
$ DEFINE/SYSTEM /EXECUTIVE_MODE RDBMONITOR JL$DISK:[NEWPORT]
%DCL-I-SUPERSEDE, previous value of RDBMONITOR has been superseded
$ ! Next, either start the monitor manually or edit and run the
$ ! RMONSTART.COM procedure to start the monitor.
$ !
$ ! To start the monitor manually, enter the following command:
$ RMU/MONITOR START/OUTPUT=RDBMONITOR:MY_MONITOR.LOG
$ !
$ ! To start the monitor by using the RMONSTART.COM procedure,
$ ! modify the RMONSTART.COM file to use the RDBMONITOR logical name
$ ! by adding the /OUTPUT=RDBMONITOR qualifier.
$ EDIT SYS$STARTUP:RMONSTART.COM
.
.
.
$ RMU/MONITOR START /OUTPUT=RDBMONITOR
.
.
.
[EOB]
* EXIT
$ @RMONSTART
```

To display the contents of the new monitor log file, enter the command shown in Example 2-6.

Example 2-6 Displaying the Contents of the Monitor Log File

```
$ TYPE RDBMONITOR:MY_MONITOR.LOG
$
$ ! Use TPU to examine a running log file.
$
$ EDIT/TPU/READ_ONLY RDBMONITOR:MY_MONITOR.LOG
```

If you define your own RDBMONITOR logical name, remember to include the new DEFINE command in the SYSTARTUP.COM file so it will be executed each time your system is started. ♦

2.1.3 Changing the Monitor Process Priority

By default, the monitor process is given a base priority of 15, the highest interactive priority possible, when it is started. This default base priority level is suitable for most environments. You can change the monitor process base priority by stopping and restarting the monitor interactively, as shown in Example 2-7.

Example 2-7 Stopping and Changing the Monitor Process Priority

```
$ RMU/MONITOR STOP
$ RMU/MONITOR START/PRIORITY=6
```

Do not specify a base priority for the Oracle Rdb monitor that is lower than the base priority for any user's process doing database activity. Giving the monitor process a base priority higher than any user process base priority ensures that the monitor is not locked out from CPU access by application programs. If the monitor process cannot get CPU time, you cannot use your database.

Note

Be sure that the monitor process is not locked during database recovery operations. If the database recovery process (DBR) created by the monitor (which has the same base priority as the monitor) is locked out from the CPU, you may not be able to recover and use your database.

2.1.4 Reopening the Monitor Log File

Use the RMU Monitor Reopen_Log command to create a new version of the monitor log file, as shown in Example 2-8 (you must have the same privileges for this operation as for starting and stopping the monitor process, as described in Section 2.1.1).

Example 2-8 Reopening the Monitor Log File

```
$ RMU/MONITOR REOPEN_LOG
```

The monitor process creates a new version of the monitor log file in the same directory and with the same name as the old version. You can use this command to enable monitor logging again if it has been disabled due to an error. Use the RMU Show System command to determine if monitor logging has been disabled, as shown in Example 2-9.

Example 2–9 Showing Oracle Rdb System Status

```
$ RMU/SHOW SYSTEM
Oracle Rdb V7.0-0 on node MOE 13-SEP-1995 10:35:09.66
  - monitor logging is disabled

database DUA01:[KIBBLES.RDMS3]CORE.RDB;3
  - 1 active database user
```

Monitor logging is disabled if an error occurs that prevents the monitor process from writing to the monitor log file. You can determine the error by looking at the operator console log. Usually the error is a device-full error that prevents the monitor from extending the log file. When that happens, the monitor disables logging but otherwise functions normally.

Use the RMU Monitor Reopen_Log command shown in Example 2–10 to enable logging again after you correct the error that caused logging to be disabled.

Example 2–10 Reopening the Monitor Log File and Showing Oracle Rdb System Status

```
$ RMU/MONITOR REOPEN_LOG
$ RMU/SHOW SYSTEM
Oracle Rdb V7.0-0 on node MOE 13-SEP-1995 10:35:09.66
database DUA01:[KIBBLES.RDMS3]CORE.RDB;3
  - 1 active database user
```

Note

Purge monitor log files periodically. They can become quite large and take up disk space.

2.1.5 Reading the Monitor Log File

Each time the monitor receives a request from a user process and carries out an action, it records the activity in the monitor log file. You can examine the contents of the current monitor log file by using the operating system TYPE command, as shown in Example 2–11.

Example 2-11 Reading the Contents of the Monitor Log File

```
$ TYPE SYS$SYSTEM:RDMMON70.LOG
```

```
-----  
13-SEP-1995 18:01:45.49 - Oracle Rdb V7.0-0 database monitor started  
-----
```

```
.  
.  
.
```

```
Current time is 13-SEP-1995 18:01:45.50
```

```
13-SEP-1995 12:04:45.64: Linked MON (RDBVMS) RDBVMS70$:[KODABLD]  
13-SEP-1995 11:01:34.51: Compiled KOD$MON (LARK) KODB$:[MON]  
13-SEP-1995 10:58:29.76: Compiled KOD$LIBRARY (LARK) KODB$:[CODE]
```

```
=====
```

```
GETSYI - System Information (SYI$_)
```

```
=====
```

```
.  
.  
.
```

```
=====
```

```
Current User Status
```

```
=====
```

```
database SYS$SYSTEM:PERSONNEL.RDB;5  
  * database is available for utility access only  
  - 1 active database user  
  
  - 21E00AB5:1 - Ramses, RAMSES - active user  
  - image _$999$DUA10:[VIEWS]SQL$.EXE;1  
database DUA01:[KIBBLES.RDMS]CORE.RDB;3  
  - 1 active database user  
  
  - 21E00894:1 - _VTA74:, ORION - active user  
  - image _$998$DUA9:[TOP]SQL$.EXE;1
```

```
-----  
14-SEP-1995 14:54:41.26 - received show request from 21E0116F:0  
  - process name NEWPORT, user NEWPORT  
  - show request completed successfully  
  
14-SEP-1995 14:55:30.65 - received user attach request from 21E0116F:1  
  - process name NEWPORT, user NEWPORT  
  - for database DUA01:[DB]PERSONNEL.RDB;1 [_$996$DUA17] (13057,2,0)  
  - for utility access  
  - cluster recovery completed successfully  
  - cluster watcher is active  
  - sending normal user attach reply to 21E0116F:1
```

(continued on next page)

Example 2–11 (Cont.) Reading the Contents of the Monitor Log File

```
14-SEP-1995 14:58:10.28 - received user image termination from 21E00AB5:1
- database shutdown of DUA02:[DB]PERSONNEL.RDB;5 is complete
13-SEP-1995 14:59:36.29 - received user image termination from 21E0116F:1
- abnormal user termination detected
- database monitor created recovery process RDM_RB23_1 (22404453)
- user termination suspended until recovery ready
13-SEP-1995 14:01:10.42 - received recovery attach from 22404453:1
- process name RDM_RB23_1, user OXFORD
- sending normal recovery attach reply to 22404453:1
13-SEP-1995 14:01:10.43 - received recovery ready from 22404453:1
- process name RDM_RB23_1, user OXFORD
- sending normal recovery ready reply to 22404453:1
13-SEP-1995 14:01:12.43 - received recovery image termination from 22404453:1
- recovery was successful
13-SEP-1995 14:01:12.59 - received recovery process termination from 22404453:1
- recovery was successful
- database shutdown of DUA01:[DB]PERSONNEL.RDB;1 is complete
```

The last section of the log file contains a record of all database monitor activity that has taken place since the log file was opened. Example 2–11 contains the following information:

- A record of an RMU Show command
- A normal user attach request
- A normal database shutdown
- An abnormal termination by the user

You can also display the monitor log file by using a text editor. For example, to edit the file using TPU, enter the command shown in Example 2–12.

Example 2–12 Using an Editor to Read the Contents of the Monitor Log File

```
$ EDIT/TPU/READ SYS$SYSTEM:RDMMON.LOG
```

The command shown in Example 2–12 copies the monitor log file into a TPU buffer that you can search for keywords, such as abnormal user process termination. (You need SYSPRV privilege to edit the monitor log file.) The log file header contains operating system and process environment information useful for debugging.

Each monitor log file records the complete operating system and process context in which the monitor is running. You can use this information to verify settings, such as the monitor process priority (JPI\$_AUTHPRI), and other process privileges or quotas. If you have problems with Oracle Rdb software, preserve your monitor log files until those problems are resolved.

The Current User Status section of the log file header records user activity at the time the log file was created. This output is the same as that displayed when you issue an RMU Show Users command.

When you type the monitor log file, you display a record of database activity that is current to the minute before you issued the operating system TYPE command. This information is useful for monitoring the nature of database activity, including attach requests, user actions, recovery processes, and so forth.

2.2 Listing Active User Information

You can use three RMU Show commands to display information about the current status of database activity on your node:

- **RMU Show System**
Lists the databases that are currently open and a summary of the number of users currently accessing each database on a particular node.
- **Show Version**
Identifies the current version of Oracle Rdb.
- **RMU Show Users**
Lists the databases that are currently open and the users currently attached to each database, including recovery routines, if any, on a particular node. This command also displays global buffer information for the node on which the RMU Show Users command is issued. It displays global buffer information for the specified database only if global buffers are enabled for that database.

Use the RMU Show System command to display information about all database activity on your computer system. The RMU Show System command displays each database name and a summary of activity for each database, as shown in Example 2-13.

Example 2-13 Showing Oracle Rdb System Status

```
$ RMU/SHOW SYSTEM
Oracle Rdb V7.0-0 on node MOE 13-SEP-1995 10:35:12.66
database DUA01:[JETSON.MAD.DICT1]DATATEST.RDB;1
  - 1 active database user
database DUA01:[KIBBLES.RDMS]CORE.RDB;1
  - 1 active database user
```

Use the RMU Show Version command to see what version of Oracle Rdb is currently running. The RMU Show Version command does not show version numbers for remote databases, only for the locally installed version of Oracle Rdb, as shown in Example 2-14.

Example 2-14 Showing the Current Version of Oracle Rdb

```
$ RMU/SHOW VERSION
Executing RMU for Oracle Rdb V7.0-0
Database DUA01:[DB]MF_PERSONNEL.RDB;1 requires version 7.0
```

You can use the SHOW VERSION statement in SQL to determine what version of Oracle Rdb is running.

You can include a database file specification with your RMU Show Users command to display active users for a single database only, as shown in Example 2-15.

Oracle Rdb displays information about the following three types of database users:

- A program or interactive terminal that is attached to the database by means of an SQL ATTACH statement
- A terminal operator or command procedure that has issued an RMU Open command
- The automatic recovery routine, which processes recovery-unit journal (.ruj) files

Example 2–15 Showing Open Databases and Attached Users

```
$ RMU/SHOW USERS MF_PERSONNEL
Oracle Rdb V7.0-0 on node MOE 13-SEP-1995 10:35:15.66

database DUA01:[DB]:MF_PERSONNEL.RDB;1
  * database is opened by an operator
  - global buffer count is 50
  - maximum global buffer count per user is 2
  - 48 global buffers free
  - 1 active database user

  - 21E0116F:1 - NEWPORT, NEWPORT - active user
    - image SYS$SYSTEM:SQL$.EXE;1
    - 2 global buffers free
```

Many processes begin and end automatically. Consequently, repeated RMU Show Users commands can produce displays that differ from one moment to the next.

2.3 Displaying Clusterwide User Information

Use the RMU Dump Users command to display a list of all database users on the cluster, as shown in Example 2–16.

Example 2–16 Displaying a List of Database Users on a Cluster System

```
$ RMU/DUMP/USERS MF_PERSONNEL
Active user with process ID 216055B5
  Stream ID is 1
  Monitor ID is 2
  Transaction ID is 21
  Recovery journal filename is DUA01:[RDM$RUJ]MF_PERSONNEL$0097232208D5D240.RUJ;1
  Read/write transaction in progress
  Transaction sequence number is 128

Active user with process ID 226008B4
  Stream ID is 1
  Monitor ID is 1
  Transaction ID is 82
  Snapshot transaction in progress
  Transaction sequence number is 129

Active user with process ID 22600839
  Stream ID is 1
  Monitor ID is 1
  Transaction ID is 162
  No transaction in progress
```

Each monitor process is assigned a monitor identification number (ID) when it establishes access to a database. Example 2–16 shows the monitor IDs for each monitor process accessing the database and the file specifications for the .ruj files. For a complete summary of activity for that database, you can execute the RMU Show Users and the RMU Show Statistics commands on each node in the cluster on which database user processes reside. To do this, you can make command procedures of the RMU commands you want to execute and submit them simultaneously to batch queues on each node. Use the Output= qualifier with the RMU commands to direct the output to disk files in a single directory to which you have access, and then display or print the contents of the files.

OpenVMS OpenVMS
VAX Alpha

For OpenVMS Version 6.0 or higher, you can use the System Management utility (SYSMAN) to centralize the management of nodes and clusters. Example 2–17 shows one way to use SYSMAN to show the users on all nodes in the cluster, in this case two nodes. You need OpenVMS OPER privilege to perform this operation. Also, if you are accessing remote (nonlocal) nodes, you may need to use a password to gain access to all nodes. See the OpenVMS system management utility documentation for more information.

Example 2–17 Using SYSMAN and the RMU Show Users Command on a Cluster Configuration

```
$ MCR SYSMAN
SYSMAN> SET ENVIRONMENT /CLUSTER /USERNAME=ASTER
Remote Password: (enter your user password here)
%SYSMAN-I-ENV, current command environment:
    Clusterwide on local cluster
    Username ASTER will be used on nonlocal nodes

SYSMAN> DO RMU/SHOW USERS
%SYSMAN-I-OUTPUT, command execution on node BUSTER
Oracle Rdb V7.0-0 on node BUSTER 13-SEP-1995 15:17:59.54
database EXAMP:[ORION]MF_PERSONNEL.RDB;1
    - 1 active user
    - 25A00944:1 - ORION, ORION - active user
      - image $100$DUAL:[SYS0.SYSCOMMON.][SYSEXE]SQL$.EXE
%SYSMAN-I-OUTPUT, command execution on node MUSTER
Oracle Rdb V7.0-0 on node MUSTER 13-SEP-1995 15:17:59.54
database CHAMP:[ORION]MF_PERSONNEL.RDB;1
    - 1 active user
    - 14B11843:1 - ALPHA, ALPHA - active user
      - image $100$DUAL:[SYS0.SYSCOMMON.][SYSEXE]SQL$.EXE
SYSMAN> EXIT
$
```

Example 2–17 shows one active user each on nodes BUSTER and MUSTER, both using the SQL interactive interface. When snapshot (.snp) files are enabled, a read/write transaction writes before-images to the snapshot file for the rows being updated, and attaches the transaction sequence number (TSN) to these rows. All database changes are tracked by the TSN. If after-image journaling is enabled, the .ruj file holds all before-images of updates based on TSNs and the after-images are written to the after-image journal (.aij) file based on TSNs.

If a system problem occurs and the database becomes corrupt, the database must be restored to its previous state just prior to the time of the failure before users can attach to it again. The .ruj file automatically rolls back uncommitted update changes to the database when the system comes back on line. Because the .aij file logs all committed transactions since the previous backup operation of the database (or since the after-image journaling was enabled), the .aij file can roll forward committed changes to a database restored from a previous backup operation, if necessary. Both processes make modifications to the database based on the TSNs of each transaction for the .ruj file of each active process, and for all committed transactions in the single .aij file.

The TSN is used to schedule the use of database resources for each active transaction that needs to lock tables, index node records, and data rows. This scheduling is important in a multiuser environment to carry out transactions in an orderly manner.

2.4 Displaying Database Characteristics

Database characteristics describe the current settings of Oracle Rdb options, for example, the number of database users and the current state of certain database operations, such as backup and restore.

When you create an Rdb database, you establish database characteristics either explicitly or by default. You can modify some database characteristics by using the SQL ALTER SCHEMA statement.

To determine the current settings, use the RMU Dump or the RMU Dump Header command to display database characteristics contained in the database root file. Either command displays database header information such as database parameters, storage area parameters, snapshot area for storage area parameters, and user information, such as the number of active users.

For example, the following RMU Dump Header command displays the mf_personnel database header file information on a Digital UNIX system:

```
# rmu -dump -header mf_personnel
```

If you created the sample mf_personnel database as explained in Chapter 1, you can enter this command yourself and read the complete output.

Security Auditing on OpenVMS

OpenVMS OpenVMS
VAX Alpha

This chapter describes the security auditing performed by Oracle Rdb running on OpenVMS systems.

Security auditing a database provides a means of easily recognizing attempts to compromise the security of a database. When you invoke security auditing, you can monitor and collect security audit records on many specific operations performed on a database. You can also send alarms to specified security operators' terminals notifying security operators of significant events as they happen. By periodically reviewing security audit records in a process called audit analysis, you can determine how secure a particular database is and decide if further steps are necessary to make it more secure.

Oracle Rdb security auditing is modeled after the OpenVMS auditing model and uses several components of this model. This chapter briefly describes those aspects of OpenVMS security auditing that are used by Oracle Rdb. Refer to the OpenVMS document set for a more detailed description of OpenVMS security auditing and how it works.

This chapter describes the Oracle Rdb implementation of security auditing as it relates to Oracle Rdb objects. The chapter is divided into four sections that provide the following information:

- An overview of Oracle Rdb security auditing, see Section 3.1
- A description of the security audit event types, see Section 3.2
- How to define security events to be audited, see Section 3.3
- How to review security audit information, see Section 3.4

3.1 An Overview of Oracle Rdb Security Auditing

Oracle Rdb security auditing records security-relevant events as they occur on a per-database basis. A security-relevant event is some operation that affects the security of the objects in the database. Security-relevant events are characteristically grouped into these event types: Audit, Daccess (discretionary access), Protection, and RMU. See Section 3.2 for descriptions of these event types. Three types of objects can be audited: Database, Table, and Column.

3.1.1 Default Security Auditing and Oracle RMU Security Auditing Commands

By default, Oracle Rdb security auditing is enabled for the Audit event type. When you start security auditing, all uses of the RMU Set Audit command are audited.

Use the RMU Set Audit command to start or stop security auditing, to enable or disable security auditing for different event types, to change event types currently being audited, or to modify any audit features currently enabled for an Oracle Rdb database. The RMU Set Audit command is the Oracle Rdb equivalent to the DCL command, SET AUDIT.

Use the RMU Show Audit command to display the current set of security auditing characteristics in effect. The RMU Show Audit command is the Oracle Rdb equivalent to the DCL command, SHOW AUDIT.

Use the RMU Load Audit command to load database security audit records from the binary OpenVMS security audit journal file into an Oracle Rdb table. The RMU Load Audit command is the Oracle Rdb equivalent to the DCL command ANALYZE AUDIT. Both commands transform the binary information into formatted ASCII text for further analysis. See the *Oracle RMU Reference Manual* for a complete description of these Oracle Rdb security audit commands.

Section 3.3 describes how to use the RMU Set Audit and RMU Show Audit commands, set security auditing characteristics, and show the current security auditing settings. Section 3.4 describes how to use the RMU Load Audit command to create an Oracle Rdb table and how to review security audit information.

3.1.2 Use of OpenVMS Security Auditing

Oracle Rdb security auditing uses several components of the OpenVMS Audit utility. The most important component is the OpenVMS audit server. The OpenVMS audit server performs the following functions:

- Handles the processing and distribution of all audit information
- Monitors security auditing resources (virtual memory and disk space)
- Prevents the loss of security information when resources are depleted

The OpenVMS audit server performs its tasks in the following order:

1. Receives a binary-formatted audit packet message in the OpenVMS audit server mailbox (MBA3)
2. Writes a copy of the binary message to the security archive file, if enabled, and to the OpenVMS security audit log file
3. Formats the binary message into ASCII text and sends the security alarm message to the operator communication manager (OPCOM) process that displays the message at the security operator terminals

Oracle Rdb generates and populates its own audit packets, and sends the binary audit messages to the OpenVMS audit server by using the MBA3 mailbox. The OpenVMS audit server stores Oracle Rdb security audit records in the OpenVMS security audit journal file and relays security alarm messages to the security operator terminals.

Oracle Rdb security audit information coexists with OpenVMS audit information in the OpenVMS security audit log file. You can use the DCL command, `SHOW AUDIT/JOURNAL`, to determine the security audit journal file currently being used by your database. The `RMU Load Audit` command can extract Oracle Rdb information from the OpenVMS security audit log file for a particular database and place it in an Oracle Rdb table for further analysis.

Security auditing events for an Oracle Rdb database can result in two forms of output: alarms and audits. Alarms are one-time notifications of Oracle Rdb database security events that are sent to all terminals enabled as security operators. Audits are the audit records or recorded history of Oracle Rdb security events that are written to the OpenVMS audit journal file.

3.1.3 Monitoring Security Auditing Resources

The audit server process monitors system resources and alerts you if resource limitations develop that threaten the orderly processing of security auditing messages. The following auditing resource problems might be encountered:

- The volume of messages being written to the OPCOM mailbox exceeds the capacity of the mailbox.
- The disk that holds the system security log file has no more free disk space.
- Virtual memory for the audit server process is not sufficient.
- The logical link connection to a remote node (containing the security archive file) is broken.

See the OpenVMS documentation set for a detailed description of these problems and how each can be resolved. Because Oracle Rdb security auditing uses the same system resources as OpenVMS security auditing, you need to know where to check if an auditing resource problem develops.

3.2 Security Audit Event Types

Four security-relevant events can occur in Oracle Rdb. Each event type records information in a unique format and addresses different auditing needs within Oracle Rdb. The event types are:

- Audit—Describes the auditing of audit changes.
- Daccess—Describes the auditing of object access, using privileges.
- Protection—Describes the auditing of privilege set modification.
- Oracle RMU—Describes the auditing of Oracle RMU events.

Sections 3.2.1, 3.2.2, 3.2.3, and 3.2.4 describe each event type.

3.2.1 The Audit Event Type

The Audit event type describes the Audit events that occur when you issue the RMU Set Audit command to indicate that the auditing (and security) characteristics for the database have changed. Security alarms and security audit journal records are both produced by default for this audit event type. The auditing information recorded for the RMU Set Audit command notes the modifications performed to the audited database if the results of the RMU Set Audit command are successful. Unsuccessful Audit events are not audited because they do not directly affect the security or auditing of the database.

Audit is the default event type. You cannot disable Audit with the RMU Set Audit command. Audit is enabled when Oracle Rdb security auditing is started.

3.2.2 The Daccess Event Type

The Daccess (discretionary access) event type describes the Daccess events that occur whenever any SQL statement or Oracle RMU command is used that requires an access privilege set (APS) check. Distinct Oracle Rdb Daccess events (that is, separate SQL statements and Oracle RMU commands) can be grouped together by the privilege that is needed to execute them. Multiple SQL statements and Oracle RMU commands require the same privilege.

Oracle Rdb audits the SQL statements and Oracle RMU commands that require Oracle Rdb to make access control decisions based on privileges associated with the database objects. Oracle Rdb Daccess event auditing is therefore tightly integrated with the discretionary access control (DAC) privilege system. When the privilege is checked, a Daccess event type audit record is produced that shows the following information:

- The statement or command given
- The access required
- The privilege (OpenVMS or Oracle Rdb) used to gain access
- If the access check for the statement or command performed resulted in Success or Failure

Daccess event auditing is strictly under the control of the Oracle Rdb security administrator. The Oracle Rdb security administrator can audit user access by using any privilege to any object. This can be done by enabling the auditing of the Daccess event type, and specifying the privileges, objects, and users to be audited. In addition to the regular set of privileges that can be specified for auditing with this event type, two privileges, Success and Failure, can also be specified. These two privileges provide a quick way to indicate that all successful and failed access attempts to the database objects should be audited. See the *Oracle Rdb7 Guide to Database Design and Definition* for a list of privileges required to perform SQL data definition language (DDL) and data manipulation language (DML) operations and Oracle RMU commands.

OpenVMS has a comparable event to Daccess, the FILE_ACCESS event type. The OpenVMS system security administrator can require OpenVMS access check auditing by enabling the FILE_ACCESS event, regardless of the presence of owner-specified access mode auditing.

3.2.3 The Protection Event Type

The Protection event type describes the Protection events that occur whenever the SQL GRANT and REVOKE statements are used to change the access privilege set (APS) for a database object. The audit information recorded for these statements notes the success of the operation and the modifications performed to the APS. This procedure is different from the Daccess event type, which records only the Success of object access. Unsuccessful Protection events are not audited because they do not directly affect the security or auditing of the database. For instance, a user can enter the following SQL statement:

```
SQL> GRANT INSERT
cont> ON SCHEMA AUTHORIZATION TEST
cont> TO [RDB];
```

If auditing of the DBCTRL database privilege is enabled, then this statement results in a Daccess audit record indicating that a user attempted to access the Test database, using the DBCTRL privilege. However, unless Protection auditing is enabled, no record of the modifications is made to the APS.

3.2.4 The RMU Event Type

The RMU event type describes the Oracle RMU events that occur whenever an Oracle RMU command is performed. The auditing information notes the Oracle RMU command performed. Table 3–1 shows the Oracle RMU functions that are not audited because they cannot attach to the database to perform the auditing function.

Table 3–1 Oracle RMU Functions That Are Not Audited

Oracle RMU Function	Oracle RMU Privilege	OpenVMS Privilege
Convert database	RMU\$CONVERT RMU\$RESTORE	
Restore database	RMU\$RESTORE	READ access to the database backup (.rbf) file
Restore only root file	RMU\$RESTORE	READ access to the .rbf file
Recover database	RMU\$RESTORE	
Dump the export file		READ access to the interchange (.rbr) file

(continued on next page)

Table 3–1 (Cont.) Oracle RMU Functions That Are Not Audited

Oracle RMU Function	Oracle RMU Privilege	OpenVMS Privilege
Extract database	RMU\$UNLOAD	
Start database monitor ¹		
Reopen database monitor log ¹		
Stop database monitor ¹		
RMU Show with no database specified		WORLD
RMU Show System	RMU\$SHOW	
RMU Show Users	RMU\$BACKUP RMU\$OPEN	
Show locks		WORLD
Dump journals or backup file	RMU\$DUMP RMU\$BACKUP RMU\$RESTORE	OpenVMS access to the recovery-unit journal (.ruj) file
Optimize .ajj file	RMU\$BACKUP RMU\$RESTORE	

¹Start or stop the monitor, or reopen the monitor log file from a SYSTEM account with SETPRV privilege. The process that starts the monitor attempts to give the monitor all privileges; the required privileges are: ALTPRI, CMKRNL, DETACH, PSWAPM, SETPRV, SYSGBL, SYSNAM, SYSPRV, and WORLD.

3.3 Defining Security Events to Be Audited

To define and implement security auditing requirements for a database, you define auditing requirements at four levels of increasing scope. These four levels are:

- The user level
Use the Enable or the Disable=Identifiers=(Identifier-List) qualifier to enable or disable auditing of user access to database objects.

- **The Daccess level**
Use the Enable or the Disable=Daccess=[Object-Type=(object-name-list)] Privileges=(priv-list) qualifier to enable or disable auditing of access to database objects by users who have the privileges you specify. You can use the Type qualifier to specify that events enabled at this level produce either alarms or audit records, or both.
- **The event level**
Use the Enable or the Disable qualifier to enable or disable security auditing for events. You can use the Type qualifier to specify that events enabled at this level produce either alarms or audit records, or both.
- **The top level**
Use the Start and Stop qualifiers for starting and stopping security auditing. Use the Type=(Alarm | Audit) qualifier to specify the type of auditing to use, sending either alarms or recording audit records, or both.

Based on these four levels, Oracle Rdb recommends the following method for setting up database security auditing:

1. Design and implement security auditing at the bottom or user-level first, and progress to the top level where security auditing is actually started.
2. After defining the user-level auditing requirements, determine the discretionary access or Daccess audit event requirements and define them.
3. Determine the event-level requirements and define them. Once you have defined these three lower levels, the entire security auditing design is defined for your database.
4. Start security auditing and specify the types of auditing desired: sending alarms, recording audit records, or both.

When you use this four-level approach, everything you want audited is audited the moment you start security auditing. If you choose to implement security auditing in some other way (for example, in reverse order) your security auditing requirements are not complete the moment you start security auditing because only the default characteristics are used. As a result, events that should be audited might not be. Eventually, as you define your security auditing requirements, these items are audited, but not until you actually specify or define them as part of your security auditing system.

Setting up your auditing system in the recommended manner is especially important for the security of those databases you want audited. It is also the best way to guarantee that any or all database activity is being audited the moment you open the database for user access.

Sections 3.3.1, 3.3.2, 3.3.3, and 3.3.4 show examples of how to implement security auditing and display the results.

3.3.1 Setting User-Level Information for Security Auditing

The first level of security that should be implemented for a database is user-level security auditing. At this level, you can enable or disable auditing of user access to objects listed in the Enable or the Disable=Daccess=Object-Type qualifier. When this level of auditing is established, any user whose identifier you specify is audited for accessing the database objects with the privileges specified.

You can specify user identification code (UIC) identifiers, general identifiers, and system-defined identifiers in the identifier list. You can also use wildcard characters within the identifiers to identify groups of users. For example, the [*,*] identifier indicates public and means all users are audited. When you specify more than one identifier, separate the identifiers with commas and enclose the identifier list within parentheses. Any use of UIC identifiers with commas, such as [RDB,JONES], must be enclosed within quotation marks, as in "[RDB,JONES]". There is no order dependence in the list of identifiers. As long as the user matches some identifier anywhere in the list, that user is audited. When you use IDENT=(identifier-list) to specify one or more identifiers to be audited, those identifiers are audited whenever they access any object for which auditing has been enabled.

The following example establishes auditing for two users and then shows how to display the enabled identifiers:

```
$ RMU/SET AUDIT/ENABLE=IDENTIFIERS=("[RDB,BOB]", "[RDB,MACK]") MF_PERSONNEL
$ RMU/SHOW AUDIT /IDENTIFIERS MF_PERSONNEL
```

```
Enabled identifiers:
  (IDENTIFIER=[RDB,BOB])
  (IDENTIFIER=[RDB,MACK])
```

3.3.2 Setting Audit Access to Database Objects (Daccess Auditing)

After you have established user-level auditing, the next step is to define audit access to database objects, known as Daccess event auditing. This step allows you to audit access to database objects by users who have the privileges you specify. You can specify more than one object type (Database, Table, or Column) in a single RMU Set Audit command. If you want to establish audit access to more than one of these object types, you can specify the audit access for each in the same command. The privileges specified, however, apply to all objects specified in the same command. To ensure that the proper privileges are specified for an object, specify audit access to different objects in separate commands.

If you specify an object type, you must also specify one or more object names, and one or more privileges. If you specify more than one object name, separate the object names with commas and enclose the list within parentheses. For example, if you want to specify auditing access to the object type Table, you can then specify the list of tables you want audited, such as EMPLOYEES and JOB_HISTORY.

For the object types you specify, you must select which privileges to audit by using the Privileges=Privilege qualifier. The privileges that can be specified with the Privileges qualifier are listed in Table 3–2. Specifying a privilege for an object with an “N” in Table 3–2 does not yield an error. Because the specified privilege is never checked for that object, it is not audited.

Table 3–2 Daccess Privileges for Database Objects

SQL Privilege	DATABASE	TABLE/VIEW	COLUMN
ALTER	Y	Y	N
CREATETAB	Y	Y	N
DBADM	Y	N	N
DBCTRL	Y	Y	N
DELETE	N	Y	N
DISTRIBTRAN	Y	N	N
DROP	Y	Y	N
INSERT	N	Y	N
OPERATOR	Y	N	N
REFERENCES	N	Y	Y
SECURITY	Y	N	N
SELECT	Y	Y	N
UPDATE	N	Y	Y
ALL	Y	Y	Y
FAILURE	Y	Y	Y
SUCCESS	Y	Y	Y

Daccess event auditing does not begin immediately after an RMU Set Audit Enable=Daccess=Object-Type command is issued, nor does it end immediately after an RMU Set Audit Disable=Daccess=Object-Type command is issued. If a user has attached to a database and has accessed the audited object before the RMU Set Audit command is given, then access to that object may or may not

be audited for the life of the attach, depending on the state of auditing at the time the object was first accessed.

For any users already attached to the database, auditing does not begin after an RMU Set Audit Enable=Daccess=Object-Type command is issued until they detach and attach to the database again, then access the object whose auditing was enabled. The same is true if auditing of an object is disabled or ended. Auditing ends only for users who have detached and attached to the database again, then accessed the object whose auditing was disabled.

The following example establishes Daccess auditing for the Database object type. Users who use the Select privilege are audited. The following OpenVMS example also shows how to display the Daccess event auditing settings for a database.

```
$ RMU/SET AUDIT/ENABLE=DACCESS=DATABASE/PRIVILEGES=(SELECT) MF_PERSONNEL
$ RMU/SHOW AUDIT/DACCESS=DATABASE MF_PERSONNEL
Security auditing STOPPED for:
    DACCESS (disabled)
        DATABASE
        (SELECT)

Security alarms STOPPED for:
    DACCESS (disabled)
        DATABASE
        (SELECT)
```

The following example establishes Daccess auditing for the Table object type. The example shows the EMPLOYEES and JOB_HISTORY tables for users who use the Insert, Update, and Select privileges. This example also shows how to display the Daccess event auditing settings for tables. By default, both alarms and audit records are enabled.

```
$ RMU/SET AUDIT/ENABLE=DACCESS=TABLE=(EMPLOYEES,JOB_HISTORY) -
_$_ /PRIVILEGES=(INSERT,UPDATE,SELECT) MF_PERSONNEL
$ RMU/SHOW AUDIT /DACCESS=TABLE MF_PERSONNEL
Security auditing STOPPED & enabled for:
    DACCESS (disabled)
        TABLE : EMPLOYEES
        (SELECT,INSERT,UPDATE)
        TABLE : JOB_HISTORY
        (SELECT,INSERT,UPDATE)

Security alarms STOPPED & enabled for:
    DACCESS (disabled)
        TABLE : EMPLOYEES
        (SELECT,INSERT,UPDATE)
        TABLE : JOB_HISTORY
        (SELECT,INSERT,UPDATE)
```

The following example establishes Daccess auditing for the Column object type. The example shows the EMPLOYEE_ID column of the EMPLOYEES table and the COLLEGE_CODE column of the COLLEGES table for users who use the Select privilege. This example also shows how to display the Daccess event auditing settings for columns. By default, both alarms and audit records are enabled.

```
$ RMU/SET AUDIT/ENABLE=DACCESS=COLUMN= -
_$(EMPLOYEES.EMPLOYEE_ID,COLLEGES.COLLEGE_CODE) -
_$/PRIVILEGES=(UPDATE) MF_PERSONNEL
$ RMU/SHOW AUDIT/DACCESS=COLUMN MF_PERSONNEL
Security auditing STOPPED for:
  DACCESS (disabled)
    COLUMN : COLLEGES.COLLEGE_CODE
      (UPDATE)
    COLUMN : EMPLOYEES.EMPLOYEE_ID
      (UPDATE)
Security alarms STOPPED for:
  DACCESS (disabled)
    COLUMN : COLLEGES.COLLEGE_CODE
      (UPDATE)
    COLUMN : EMPLOYEES.EMPLOYEE_ID
      (UPDATE)
```

3.3.3 Enabling and Disabling Event Information for Security Auditing

The third step in setting up your security auditing scheme is to establish the events (Audit, Daccess, Protection, or Oracle RMU) that you want audited. Of these four events, only Daccess, Protection, and Oracle RMU events can be enabled or disabled. The Audit event is always enabled. Auditing of these events begins immediately for all users, including those attached to the database.

The following example shows how to enable Daccess, Protection, and Oracle RMU event auditing and how to display the security auditing settings. By default, both alarms and audit records are enabled.

```
$ RMU/SET AUDIT/ENABLE=(DACCESS,PROTECTION,RMU) MF_PERSONNEL
$ RMU/SHOW AUDIT /ALL MF_PERSONNEL
Security auditing STOPPED & enabled for:
  PROTECTION (enabled)
  RMU (enabled)
  AUDIT (enabled)
  DACCESS (enabled)
Security alarms STOPPED & enabled for:
  PROTECTION (enabled)
  RMU (enabled)
  AUDIT (enabled)
  DACCESS (enabled)
```

```
Audit flush is disabled
Audit every access
Enabled identifiers:
  (IDENTIFIER=[RDB,BOB])
  (IDENTIFIER=[RDB,MACK])
```

3.3.4 Starting and Stopping Security Auditing and Other Auditing Characteristics

The last step in setting up your security auditing scheme is to set the following auditing characteristics before starting security auditing:

- Decide if every access attempt or just the first access attempt is to be audited for Daccess audit event objects.
- Decide if each audit record is to be flushed individually to the security audit file or if audit records will be buffered and written to the security audit file according to a specified time interval.
- Select the type of auditing desired: alarms, audit records, or both.

To audit every access attempt or just the first access attempt to objects specified for the Daccess audit event, specify the RMU Set Audit command with the Every or the First qualifier, respectively. Both qualifiers apply to all Daccess audit event objects for the database.

The Every qualifier applies to the Daccess event for all database objects specified by Enable=Daccess=*Object-Type* qualifier. It is used to specify the frequency of production of alarms and audit records for the Daccess event. If the Every qualifier is specified, then an alarm or audit record is produced each time an access check is performed for an object specified by the Enable=Daccess clause.

If the First qualifier is specified, then during an attach, only the first time the user accesses an object with a privilege does the First qualifier produce an alarm or audit record. This is useful because Daccess auditing is closely tied to access (privilege) checking in Oracle Rdb. A user's access to an object is determined at the time that the user first accesses the object during an attach. This means that during an attach, object access never changes (the user always has the same set of privileges for an object during an attach).

If the user successfully accesses an object with the appropriate privilege once during an attach, then all subsequent accesses to that object with the same privilege are also successful. The same is true for failed access. Therefore, auditing using the Every qualifier, where every access with the same privilege is audited, may yield some redundant auditing information. You can use

the Every qualifier for gathering statistics about the frequency of access to database objects.

The following example specifies every enabled, auditable event be audited. This example also shows how to display the audit settings to determine if every access of an auditable event is set.

```
$ RMU/SET AUDIT/EVERY MF_PERSONNEL  
$ RMU/SHOW AUDIT/EVERY MF_PERSONNEL
```

Audit every access

The following example specifies that just the first enabled, auditable event be audited. This example also shows how to display the audit settings to determine if just the first access of an auditable event is set.

```
$ RMU/SET AUDIT/FIRST MF_PERSONNEL  
$ RMU/SHOW AUDIT/EVERY MF_PERSONNEL
```

Audit first access only

You can use the Flush qualifier with the RMU Set Audit command to flush each audit record as it occurs. The audit record is flushed from the buffer to the security audit file. The Noflush qualifier permits audit records to be buffered for a period of time before the records are written to the security audit file. The following example enables flushing and shows that flushing is enabled for a database auditing scheme:

```
$ RMU/SET AUDIT/FLUSH MF_PERSONNEL  
$ RMU/SHOW AUDIT/FLUSH MF_PERSONNEL
```

Audit flush is enabled

The following example disables flushing and also shows that flushing is disabled for a database auditing scheme:

```
$ RMU/SET AUDIT/NOFLUSH MF_PERSONNEL  
$ RMU/SHOW AUDIT/FLUSH MF_PERSONNEL
```

Audit flush is disabled

When you set alarms or set audit records, subsequent qualifiers in the command line (Start, Stop, Enable, Disable) generate or affect security alarm messages and audit records. The alarm messages are sent to all terminals enabled as security operator terminals. The security audit journal records are recorded in the security audit journal. You specify the security audit journal using the DCL command, Set Audit Journal Destination.

By default, both security alarms and security audit journals are enabled. If you do not specify the Type qualifier within the RMU Set Audit command, Oracle RMU enables or disables both security alarms and security audit journals, depending on whether the Enable or Disable qualifier is specified. You must specify the Type qualifier with the Start, Stop, Enable, or Disable qualifier for it to work. You cannot specify both the Alarm and Audit keywords within the same Type qualifier clause on the command line. However, you can specify each keyword separately within separate Type qualifier clauses on the same command line or in two different commands. You can also accept the default of both alarms and audit records being enabled.

The following example shows how to specify the type of auditing desired, how to start security auditing, and how to display the security audit settings:

```
$ RMU/SET AUDIT /TYPE=AUDIT /START MF_PERSONNEL
$ RMU/SHOW AUDIT /ALL MF_PERSONNEL
Security auditing STARTED & enabled for:
  PROTECTION (enabled)
  RMU (enabled)
  AUDIT (enabled)
  DACCESS (enabled)

Security alarms STOPPED & enabled for:
  PROTECTION (enabled)
  RMU (enabled)
  AUDIT (enabled)
  DACCESS (enabled)

Audit flush is disabled

Audit every access

Enabled identifiers:
  (IDENTIFIER=[RDB,BOB])
  (IDENTIFIER=[RDB,MACK])
```

If you want to enable alarms and show that they are enabled, enter the following commands:

```
$ RMU/SET AUDIT /TYPE=ALARM /START MF_PERSONNEL
$ RMU/SHOW AUDIT /ALL MF_PERSONNEL
Security auditing STARTED & enabled for:
  PROTECTION (enabled)
  RMU (enabled)
  AUDIT (enabled)
  DACCESS (enabled)

Security alarms STARTED & enabled for:
  PROTECTION (enabled)
  RMU (enabled)
  AUDIT (enabled)
  DACCESS (enabled)
```

```
Audit flush is disabled
Audit every access
Enabled identifiers:
  (IDENTIFIER=[RDB,BOB])
  (IDENTIFIER=[RDB,MACK])
```

3.4 Reviewing Security Audit Information

Security audit information exists in two forms, alarm messages and audit records. If both alarms and security audit journaling are in effect, you can obtain a record of the security event through a hardcopy printout or the audit journal file.

Alarm messages are short-lived, as they display on the screen until replaced by other alarm or system messages on a security operator's terminal. For this reason, you should select a terminal that provides a hardcopy printout to capture alarm information. Alarm messages are described in more detail in Section 3.4.1.

The usefulness of the security audit journal depends upon the procedures you adopt to review the file on a regular basis. For example, you might choose to create a new OpenVMS security log file each morning and review the previous day's version for suspicious activity. Consider these dependencies:

- The number of security events that you are auditing for a particular Oracle Rdb database
- How many databases are being audited

Then determine if it is practical to review every audited record written to the audit journal file. As an alternative, you might want to select a specific set of records from the journal file, for example, all events that required certain privileges to perform a task, or all events created outside normal business hours. If you find any security events that appear suspicious, you can inspect the security audit journal file more thoroughly. If you find no events of a suspicious nature, you can archive the file for permanent storage of all security audit records, including the Oracle Rdb database security auditing records.

The procedures for loading and reading audit journal records are described in Section 3.4.2 and Section 3.4.3, respectively.

3.4.1 Interpreting Security Auditing Alarm Information

When you set audit access to a database object as in Table 3–2 for the `EMPLOYEES` and `JOB_HISTORY` object names, any access to those object names sends an alarm message to the security operator's terminal. Object name access depends upon the privilege specifications in effect.

To set up your terminal as a security operator's terminal, you will need to perform the following three steps:

1. Be sure you are assigned the OpenVMS `SETPRV` and `SECURITY` privileges, so you can perform the operations to make your terminal a security operator's terminal.
2. Make your terminal a security operator's terminal by entering the following command:

```
$ REPLY/ENABLE=SECURITY
```

The standard information contained in every alarm message consists of the following:

- Timestamp of the event
- System ID
- Database name
- Auditable event name
- Process identification (PID) number
- Event time
- User's name
- User's node name
- Type of auditable event (if an RMU event, then the phrase displays with the actual Oracle RMU command)
- Object name (omitted if an RMU event)
- Object type (omitted if an RMU event)
- Type of operation performed (omitted if an RMU event)
- Access requested
- Substatus of the operation performed
- Final outcome of the event

- Oracle Rdb privilege used in the operation

This information is shown in more detail in Sections 3.4.1.1, 3.4.1.2, 3.4.1.3, and 3.4.1.4.

3.4.1.1 Interpreting AUDIT Event Alarm Information

Once you have set up your terminal as a security operator's terminal, security audit alarm messages are displayed or printed out. For example, the RMU Set Audit command displays the following successful alarm message:

```
$ RMU/SET AUDIT/ENABLE=DACCESS=DATABASE/PRIVILEGES=(SELECT) MF_PERSONNEL
$
%%%%%%%%%% OPCOM 26-AUG-1995 15:47:56.60 %%%%%%%%%%% (from node
STORMY at 26-AUG-1995 15:48:12.92)
Message from user MACK on STORMY
Oracle Rdb Security alarm (SECURITY) on STORMY, system id: 32122
Database name:          DUA1:MF_PERSONNEL.RDB;1
Auditable event:       Auditing change
PID:                   3B807EE3
Event time:            26-AUG-1995 15:47:56.50
User name:             MACK
RMU command:           RMU/SET AUDIT/ENABLE=DACCESS=DATABASE
                       /PRIVILEGES=(SELECT) MF_PERSONNEL
Access requested:      RMU$SECURITY
Sub status:            RMU required privilege
Final status:          %SYSTEM-S-NORMAL
RMU privilege used:    RMU$SECURITY
```

This alarm message shows an Audit event type change.

3.4.1.2 Interpreting Daccess Event Alarm Information

To show what alarm messages display for Daccess event types, access the EMPLOYEES table of the mf_personnel database with the following SQL statements:

```
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES;
```

The following OPCOM message displays on your terminal screen because you have printed the EMPLOYEES rows, using the audited SELECT (SQL SELECT) privilege:

```
%%%%%%%%%% OPCOM 26-AUG-1995 17:28:57.92 %%%%%%%%%% (from node
STORMY at 26-AUG-1995 17:29:12.92)
Message from user MACK on STORMY
Oracle Rdb Security alarm (SECURITY) on STORMY, system id: 32601
Database name:          DUA1:MF_PERSONNEL.RDB;1
Auditable event:       Attempted table access
PID:                   3C215052
Event time:            26-AUG-1995 17:29:12.88
User name:             MACK
Object name:           EMPLOYEES
Object type:           TABLE
Operation:             Select Record
Access requested:      SELECT
Sub status:            Oracle Rdb required privilege
Final status:          %SYSTEM-S-NORMAL
Oracle Rdb privilege used: SELECT
```

The message provides the following information:

- **Timestamp of the event (26-AUG-1995 17:28:57.92)**
- **System ID (32601)**
- **Database name (DUA1:MF_PERSONNEL.RDB;1)**
- **Auditable event (Attempted table access)**
- **Process identification (PID) number (3C215052)**
- **Event time (26-AUG-1995 17:29:12.88)**
- **User's name (MACK)**
- **User's node name (STORMY)**
- **Object name (EMPLOYEES)**
- **Object type (TABLE)**
- **Type of operation performed (Select Record)**
- **Access requested (SELECT)**
- **Substatus of the operation performed (Oracle Rdb required privilege)**
- **Final outcome of the event (%SYSTEM-S-NORMAL)**
- **Oracle Rdb privilege used in the operation (SELECT)**

3.4.1.3 Interpreting Protection Event Alarm Information

To show the alarm message that displays for Protection event types, perform an SQL GRANT statement on the mf_personnel database with the following SQL statement:

```
SQL> GRANT DELETE ON EMPLOYEES TO [RDB,MACK] WITH GRANT OPTION;
```

The following Protection alarm message displays:

```
$
%%%%%%%%%% OPCOM 26-AUG-1995 16:54:34.76 %%%%%%%%%%% (from node
STORMY at 26-AUG-1995 16:54:56.59)
Message from user MACK on STORMY
Oracle Rdb Security alarm (SECURITY) on STORMY, system id: 32127
Database name:          DUAL:MF_PERSONNEL.RDB;1
Auditable event:       Protection change
PID:                   3F009EF8
Event time:            26-AUG-1995 16:54:56.42
User name:             MACK
Object name:           EMPLOYEES
Object type:           TABLE
Grantee:               [RDB,MACK]
New ACE privileges:    DELETE
Old ACE privileges:
Final status:          %SYSTEM-S-NORMAL
```

The final status returned was %SYSTEM-S-NORMAL and that the new ACE privilege is DELETE. This indicates a successful alarm message. The old ACE privileges field is blank because the user MACK had no prior privileges for the EMPLOYEES table.

3.4.1.4 Interpreting Oracle RMU Event Alarm Information

To show what alarm messages display for Oracle RMU event types, perform an Oracle RMU command on the mf_personnel database as follows:

```
$ RMU/SHOW AUDIT MF_PERSONNEL
```

The following successful alarm message displays:

```

$
%%%%%%%%% OPCOM 26-AUG-1995 16:36:20.95 %%%%%%%%%% (from node
STORMY at 26-AUG-1995 16:36:52.95)
Message from user MACK on STORMY
Oracle Rdb Security alarm (SECURITY) on STORMY, system id: 32601
Database name:          DUAL:MF_PERSONNEL.RDB;1
Auditable event:       Attempted RMU command
PID:                   3B807EE3
Event time:            26-AUG-1995 16:36:20.85
User name:             MACK
RMU command:           RMU/SHOW AUDIT MF_PERSONNEL
Access requested:      RMU$SECURITY
Sub status:            RMU required privilege
Final status:          %SYSTEM-S-NORMAL
RMU privilege used:    RMU$SECURITY

```

If you do not have the privilege to load the security audit records into a database file, OPCOM sends an alarm message showing the outcome of %RDB-E-NO_PRIV.

```

$ RMU/LOAD/AUDIT MF_PERSONNEL AUDIT_RECORDS -
_ $ SYS$COMMON:[SYSMGR]SECURITY_AUDIT.AUDIT$JOURNAL
%%%%%%%%% OPCOM 26-AUG-1995 17:20:34.54 %%%%%%%%%% (from node
STORMY at 26-AUG-1995 17:20:57.97)
Message from user MACK on STORMY
Oracle Rdb Security alarm (SECURITY) on STORMY, system id: 32601
Database name:          DUAL:MF_PERSONNEL.RDB;1
Auditable event:       Attempted RMU command
PID:                   4140396B
Event time:            26-AUG-1995 17:20:57.84
User name:             MACK
RMU command:           RMU/LOAD/AUDIT MF_PERSONNEL AUDIT_RECORDS
                        SYS$COMMON:[SYSMGR]SECURITY_AUDIT.AUDIT$JOURNAL
Access requested:      RMU$SECURITY
Sub status:            RMU required privilege
Final status:          %RDB-E-NO_PRIV
RMU privilege used:    RMU$SECURITY

```

```
%RDB-E-NO_PRIV, privilege denied by database facility
```

Because the user did not have the necessary privileges to perform this operation, the alarm message displays a final status message indicating that privilege is denied.

3.4.2 Using the RMU Load Audit Command

You can use the RMU Load Audit command to load audit records from the security audit journal file into a table in your database for further analysis, storage, and eventually for archival purposes if desired. The audit journal records collected on an Oracle Rdb database can only be stored in the database from which they were collected. The Oracle Rdb table into which you load the security audit journal records is defined with the columns shown in Table 3–3. Not all columns apply to (are used by) all audit events. The audit events that apply for the column are listed.

Table 3–3 Columns for Storing Security Audit Journal Records

Column Name	SQL Data Type and Length	Description	Audit Events That Apply for the Column
AUDIT\$EVENT	CHAR 16	Audit event type	All
AUDIT\$SYSTEM_NAME	CHAR 15	Name of node on which audit event occurred	All
AUDIT\$SYSTEM_ID	CHAR 12	ID of node on which audit event occurred	All
AUDIT\$TIME_STAMP	CHAR 48	Timestamp of audit event	All
AUDIT\$PROCESS_ID	CHAR 12	ID of process causing audit event	All
AUDIT\$USER_NAME	CHAR 12	OpenVMS name of user causing audit event	All
AUDIT\$TSN	CHAR 25	Transaction sequence number (TSN) of transaction causing an audit event	All
AUDIT\$OBJECT_NAME	CHAR 255	Name of database object	DACCESS, PROTECTION
AUDIT\$OBJECT_TYPE	CHAR 12	Type of object	DACCESS, PROTECTION
AUDIT\$OPERATION	CHAR 32	Statement or command performed	DACCESS

(continued on next page)

Table 3–3 (Cont.) Columns for Storing Security Audit Journal Records

Column Name	SQL Data Type and Length	Description	Audit Events That Apply for the Column
AUDIT\$DESIRED_ACCESS	CHAR 16	Oracle Rdb privilege required for statement/command	DACCESS
AUDIT\$SUB_STATUS	CHAR 32	Substatus of audit event	DACCESS
AUDIT\$FINAL_STATUS	CHAR 32	Final status of audit event	All
AUDIT\$RDB_PRIV	CHAR 16	Oracle Rdb privilege used for object access	DACCESS
AUDIT\$VMS_PRIV	CHAR 16	OpenVMS privilege used for object access	DACCESS
AUDIT\$GRANT_IDENT	CHAR 192	OpenVMS identifiers for modified access control entry (ACE)	PROTECTION
AUDIT\$NEW_ACE	CHAR 192	New access control entry	PROTECTION
AUDIT\$OLD_ACE	CHAR 192	Old access control entry	PROTECTION
AUDIT\$RMU_COMMAND	CHAR 512	RMU command given	AUDIT, RMU

The RMU Load Audit command automatically creates a table if one does not exist. To create and load the table, determine the location of the security audit journal file by entering the following DCL command:

```
$ SHOW AUDIT/JOURNAL
List of audit journals:
  Journal name:      SECURITY
  Journal owner:    (system audit journal)
  Destination:      SYS$COMMON:[SYSMGR]
                   SECURITY_AUDIT.AUDIT$JOURNAL
  Monitoring:      free disk space
  Warning threshold: 100000 blocks
  Action threshold: 50000 blocks
  Resume threshold: 80000 blocks
```

To create and load the database table named `AUDIT_RECORDS`, enter the following Oracle RMU command:

```
$ RMU/LOAD/AUDIT MF_PERSONNEL AUDIT_RECORDS -
_ $ SYS$COMMON:[SYSMGR]SECURITY_AUDIT.AUDIT$JOURNAL
%RMU-I-DATRECSTO, 85 data records stored
```

This command loads only audit records from the journal file `SECURITY_AUDIT.AUDIT$JOURNAL` that are specific to the `mf_personnel` database. That is, the database name specified is used to identify both the audit records to be loaded and the database into which they are to be loaded. The audit journal may contain audit records for other databases, but they are not loaded.

3.4.3 Reviewing Audit Journal Records

Once you have loaded the security audit records into a database table, you can perform queries on the table and further analyze the contents of the security log file. As a security administrator you may be interested in inspecting one or more of the following:

- Certain event types
- Certain users
- Date or time range (or both) of the audited events
- Type of database object for which audit records exist
- Type of statement or command performed
- Final status of the audit event
- Protection characteristics of the access control entry (ACE)

The following SQL query selects the event type column, the object name column, the object type column, the final status column, the timestamp column, and the Oracle RMU command column. Output is ordered by the event column.

```
SQL> SELECT AUDIT$EVENT,AUDIT$OBJECT_NAME,AUDIT$OBJECT_TYPE,
cont> AUDIT$FINAL_STATUS,AUDIT$TIME_STAMP,AUDIT$RMU_COMMAND FROM
cont> AUDIT_RECORDS ORDER BY AUDIT$EVENT;
AUDIT$EVENT
  AUDIT$OBJECT_NAME
    AUDIT$OBJECT_TYPE  AUDIT$FINAL_STATUS
      AUDIT$TIME_STAMP
        AUDIT$RMU_COMMAND
AUDIT
```



```

>>
>>
>>
                                %SYSTEM-S-NORMAL
26-AUG-1995 14:23:45.56
    RMU/SET AUDIT/ENABLE=IDENTIFIERS=( [RDB,MACK] )/START MF_PERSONNEL
.
.
.
DACCESS
DUAL:[MACK]MF_PERSONNEL.RDB;1
>>
>>
>>
    DATABASE                                %SYSTEM-S-NORMAL
    26-AUG-1995 15:34:34.00
.
.
.
PROTECTION
EMPLOYEES
>>
>>
>>
    TABLE                                %SYSTEM-S-NORMAL
    26-AUG-1995 20:23:16.78
.
.
.
RMU
>>
>>
>>
                                %SYSTEM-S-NORMAL
26-AUG-1995 20:55:11.45
    RMU/SET AUDIT/ENABLE=DACCESS=COLUMN=(EMPLOYEES.EMPLOYEE_ID,
    >>COLLEGES.COLLEGE_CODE)/PRIVILEGES=(SELECT) MF_PERSONNEL
.
.
.

```

You can create command procedures that query the information in the rows by looking for unusual or unexpected data, such as activity occurring outside of normal business hours. These command procedures can help automate the analysis of security audit table data. If unusual data is found, then you can inspect the row or rows more closely.

As you gather and load each day's or week's audit records in a table for further analysis, you may want to store previously inspected rows in a read-only storage area or archive them to a security archive database. This way you can keep an historical record of the security of the database if this is important for your application. In other cases, you may simply want to delete any information that you have inspected and are certain represents normal activity.

Opening and Closing a Database

Opening and closing a database is the process of mapping and unmapping database root (.rdb) file global sections (OpenVMS) or shared memory partitions (Digital UNIX). The .rdb file contains all the database-specific information that is loaded into memory and used to operate the database. Maintenance operations that modify .rdb file characteristics require exclusive access to the database; they cannot take place when the .rdb file is mapped.

Opening a database involves the following steps:

1. Checking to make sure the .rdb file has not been moved
2. Allocating database page pool space for the .rdb file
3. Creating and mapping the .rdb file into global sections or shared memory partitions
4. Starting recovery processes if the database needs recovery
5. Initializing the .aij file, if any

Closing a database involves:

- Terminating user processes (optional)
- Unmapping global sections or shared memory partitions

Before you can use a database, it must be opened. By issuing an RMU Open command, a database administrator (DBA) process can absorb the overhead of opening the database before a user has accessed it. In addition, a DBA can use the RMU Open command to open the database for restricted access to perform specific maintenance operations and to set the global buffer parameters for the database.

Otherwise, if the database is set for automatic opening and closing, the first user on each node to invoke the database incurs the overhead of opening the database, and the last user to finish incurs the overhead of closing the database. In an environment where users frequently open and close the

database, it makes sense for the DBA to use RMU Open and RMU Close commands because the database can be opened once and left open for all users.

To ensure that your DBA process absorbs the overhead, change the automatic opening option of the database to manual by using an SQL ALTER DATABASE statement, specifying the OPEN IS MANUAL argument. Then, close the database to ensure that the database is opened manually. Now you can use the RMU Open and RMU Close commands as necessary to schedule the opening and closing of the database. Leaving a database open does not generate additional system overhead.

You can leave a database open indefinitely, unless you need to perform certain maintenance operations, in which case it must be closed. For example, you cannot delete a database with the SQL DROP DATABASE statement unless the database is closed. You should be certain that no one can access the database while you perform this task. However, you can back up a database, or restore and recover individual storage areas while the database is open and users remain attached to it by using the Online qualifier with the RMU Backup, the RMU Restore, or the RMU Recover command. Using the Online qualifier is discussed more fully in Chapters 7, 8, and 9, and in the *Oracle RMU Reference Manual*.

When you use the SQL ALTER DATABASE OPEN IS MANUAL option with the RMU Open Access=Restricted command, access to the database is limited to users with SQL DBADM privilege for the database or OpenVMS BYPASS or SYSPRV privilege. Access is limited so that maintenance operations can proceed without interference from other users.

When you shut down your system, the Oracle Rdb monitor process automatically closes all your databases.

4.1 Opening a Database

A database is opened in one of the following two ways:

- By using the RMU Open command
- By attaching to the database (using the SQL ATTACH statement)

Only users with sufficient Oracle Rdb privilege (SQL DBADM) or OpenVMS BYPASS or SYSPRV privileges can enter an explicit RMU Open command to open the database. The default action is to allow automatic opening and closing of the database. If you specify Open Is Manual, a privileged user must issue an RMU Open command before any user can access the database. You can also use the SQL ALTER DATABASE . . . OPEN IS AUTOMATIC statement to reopen the database once it has been opened manually. By using the RMU Open command, you ensure that your database will keep global

sections or shared memory partitions mapped until you use the RMU Close command to unmap them.

When you open the database by using the RMU Open command, you can also choose to restrict access to the database by specifying the Access=Restricted qualifier or set the total number of global buffers and the maximum number of global buffers per user by using the Global_Buffer=(total=I,user_limit=J) qualifier. For more information on restricting access to the database, see the *Oracle RMU Reference Manual*. For more information on setting the global buffers for a database see the *Oracle Rdb7 Guide to Database Performance and Tuning* and the *Oracle RMU Reference Manual*.

4.1.1 Using the RMU Open Command

The RMU Open command saves overhead for the users accessing the database. When you open a database with the RMU Open command or the SQL ALTER DATABASE . . . OPEN IS MANUAL statement, it remains open until you issue an RMU Close command.

Opening a database explicitly by using one of these statements is important in a cluster environment because of the overhead involved in joining and leaving a group of nodes that access the same database. If you created your database for manual opening, you must use one of these statements or the RMU Open command on each node of your cluster to permit access to the database on all nodes. See the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information about the cluster environment.

To determine if a database has been opened manually, use the RMU Show Users command. (RMU is described fully in the *Oracle RMU Reference Manual*.) You will see a message that indicates how the database was opened. For example:

```
Oracle Rdb V7.0 on node TRIxie 30-AUG-1995 10:33:23.56
database AM$DISK:[RESOURCE1.DAT]MF_PERSONNEL.RDB;1
    * database is opened by an operator
```

In Example 4-1, the mf_personnel database has been opened manually and has one active user. You can also use the PATHNAME clause to open databases specified by the data dictionary path names. Example 4-1 shows an RMU Open command that opens the mf_personnel database and all the databases specified by the data dictionary path name CDD\$TOP.TEST.

Example 4–1 Opening the Database Manually and Showing Users

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> OPEN IS MANUAL;

$ RMU/OPEN MF_PERSONNEL, CDD$TOP.TEST/PATH
$ RMU/SHOW USERS MF_PERSONNEL
Oracle Rdb V7.0 on node TRIxie 30-AUG-1995 10:35:35.23

database AM$DISK:[RESOURCE1.DAT]MF_PERSONNEL.RDB;1
  * database is opened by an operator
  - global buffer count is 50
  - maximum global buffer count per user is 2
  - 48 global buffers free
  - 1 active database user

  - 20C01F2F:0 - orion, orion - active user
    - image SYS$SYSTEM:SQL$.EXE;1
    - 2 global buffers allocated
```

If you have global buffers enabled, the RMU Show Users command also displays the total global buffer count, the maximum global buffer count per user, the number of free global buffers, and the number of global buffers allocated for each user.

If you do not issue an RMU Open command for the mf_personnel database, and no user is attached to it, the output from an RMU Show Users command shows that no active users are attached to the database, as shown in Example 4–2.

Example 4–2 Closing the Database and Showing Users

```
$ RMU/CLOSE MF_PERSONNEL
$ RMU/SHOW USERS MF_PERSONNEL
Oracle Rdb V7.0 on node TRIxie 30-AUG-1995 10:36:12.34
  - no databases are accessed by this node
```

You can examine the current monitor log file (see Chapter 2 for details about the Oracle Rdb monitor) to track open and close operations performed on your database. For example, the commands in Example 4–2 produced the following log file entries:

```
-----
30-AUG-1995 13:38:08.33 - Oracle Rdb V7.0 database monitor log file reopened
-----
```

This is a VAX 8650

```
.  
. .  
30-AUG-1995 10:44:45.23 - received open database request from 20C01F2F:0  
- process name _RTA3:, user ORION  
- for database "AM$DISK:[RESOURCE1.DAT]MF_PERSONNEL.RDB;1" [_100$DUA1] (293,5,0)  
- cluster recovery completed successfully  
- cluster watcher is active  
- sending normal open database reply to requestor  
30-AUG-1995 10:45:23.46 - received show request from 20C01F2F:0  
- process name _RTA3:, user ORION  
- show request completed successfully  
30-AUG-1995 10:46:22.78 - received user attach request from 20C01F2F:1  
- process name _RTA3:, user ORION  
- for database "AM$DISK:[RESOURCE1.DAT]MF_PERSONNEL.RDB;1" [_100$DUA1] (293,5,0)  
- sending normal user attach reply to 21E005C2:1  
30-AUG-1995 10:46:12.20 - received show request from 21E005C2:1  
- process name _RTA3:, user ORION  
- show request completed successfully  
30-AUG-1995 10:46:13.56 - received user image termination from 21E005C2:1  
30-AUG-1995 10:46:55.60 - received close database request from 20C01F2F:0  
- process name _RTA3:, user ORION  
- for database "AM$DISK:[RESOURCE1.DAT]MF_PERSONNEL.RDB;1" [_100$DUA1] (293,5,0)  
- "%RDMS-F-OPERCLOSE, database operator requested database shutdown"
```

4.1.2 Attaching to a Database

In precompiled SQL or SQL module language, you must use the **DECLARE ALIAS** statement to add a database to the implicit environment. In interactive and dynamic SQL, you must use the **ATTACH** statement to add a database to the implicit environment. The **DECLARE ALIAS** statements embedded in programs or in the **DECLARE** section of an SQL module must come before any **DECLARE TRANSACTION** or **EXECUTABLE SQL** statements. **DECLARE ALIAS** statements tell the application what databases it can compile against. See the *Oracle Rdb7 SQL Reference Manual* for more information.

If you are using the SQL precompiler or SQL module language with the following programming languages, the first executable data manipulation language (DML) statement executed (normally the **SET TRANSACTION** statement) attaches to the database unless specified otherwise:

- SQL precompiler with Ada, C, COBOL, FORTRAN, or PL/I

- SQL module language with any language that supports the calling mechanism for host language programs for executing SQL statements in an SQL module file

When you create a database with automatic opening, an SQL ATTACH statement from a user opens the database as if an RMU Open command had been issued.

Example 4-3 shows a database set for automatic opening.

Example 4-3 Changing the Database Opening to Automatic and Showing Users

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> OPEN IS AUTOMATIC;
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> $RMU/SHOW USERS MF_PERSONNEL
Oracle Rdb V7.0 on node TRIxie 30-AUG-1995 12:00:01.63

database AM$DISK:[RESOURCE1.DAT]MF_PERSONNEL.RDB;1
- global buffer count is 50
- maximum global buffer count per user is 2
- 48 global buffers free
- 1 active database user

- 21805652:1 - _RTA7:, ORION - active user
- image SYSS$SYSTEM:SQL$.EXE;1
- 2 global buffers allocated
```

Because the database was opened by an attach request, the message that states the database was opened by an operator is missing from the display in Example 4-3. This sequence produced the following entries in the monitor log file:

```
-----
30-AUG-1995 10:16:54.23 Oracle Rdb V7.0 database monitor started
-----

This is a VAX 8650
.
.
.
30-AUG-1995 13:14:46.56 - received user attach request from 21805652:1
- process name _RTA3:, user ORION
- for database "AM$DISK:[RESOURCE1.DAT]MF_PERSONNEL.RDB;1" [_100$DUA1] (293,5,0)
- sending normal user attach reply to 21805652:1
```



```
30-AUG-1995 13:14:48.32 - received show request from 21805652:1
- process name _RTA3:, user ORION
- show request completed successfully
30-AUG-1995 13:14:52.78 - received user image termination from 21805652:1
.
.
.
```

Attaching to the mf_personnel database with the SQL ATTACH statement opens the database. Exiting from SQL closes the mf_personnel database because the SQL user was the only active user. An explicit SQL DISCONNECT statement declares a database closed for the process making the request and commits all active transactions for this process when no parameter is specified.

If you close your database in order to open it using the RMU Open command, you must be careful not to abort user processes accidentally. By default, the RMU Close command terminates all active processes of users on a particular node of the specified database. See Section 4.2 for more information on closing databases.

4.2 Closing a Database

The RMU Close command controls the process of eliminating active users in specific database and also unmaps global sections on OpenVMS systems or shared memory partitions on Digital UNIX systems. You can specify whether users on a single node or users on all nodes in a cluster are to be prevented from accessing the database.

The RMU Close command closes a database whether it was opened using an RMU Open command or by an attach request to the database. In either case, you must be careful not to abort active database users accidentally. By default, an RMU Close command terminates all user processes currently accessing the database you specify on that particular node. If you do not want to terminate active user processes immediately, use the Noabort qualifier with the RMU Close command. With the Noabort qualifier, users who are in the middle of a transaction when you issue the RMU Close Noabort command will not be forced off immediately, but instead will be allowed to finish their transaction. See Section 4.2.2 for more information.

When you use the RMU Close command to close a database, it does not inform users that the database is closed. Users may receive an "SYS-F-ACCONFLICT" error message when they try to access a closed database but may not realize the database was closed. However, the monitor log file records this event. Inform your user community prior to the actual closing of the database.

On OpenVMS systems, if you do want to terminate active user processes, the following commands force an exit, using Oracle Rdb:

- `RMU Close Abort=Delprc Nocluster`—Immediately forces all active users off a particular node by deleting their processes, and then closes the database.
- `RMU Close Abort=Delprc Cluster`—Immediately deletes the processes of existing users of a particular database throughout the cluster, and then closes the database.

The Cluster qualifier does not actually shut down the database on the entire cluster, but returns an error message if the database is not being used on the node from which the command is issued.

In both cases, using the `Abort=Delprc` qualifier deletes the active process and any subprocesses. The recovery-unit journal (.ruj) files are not recovered but are left in the directories to be recovered with the next attach to the database.

To ensure that .ruj files are automatically recovered when the `Abort=Delprc` qualifier is specified with the Cluster qualifier, include the Wait qualifier. When the Wait qualifier is specified with the `RMU Close` command, the Cluster qualifier actually shuts down the database for the entire cluster, even if no other users are attached to this node. The Wait qualifier also causes the operation to stall until the database is actually closed and recovered. See the *Oracle RMU Reference Manual* for more information on the Wait qualifier. When the `NoWait` qualifier is specified or the Wait qualifier is omitted, the database is closed but recovery-unit journal files are not automatically recovered. Hence, the database is unavailable to users until all recovery-unit journal files are recovered. Under this circumstance, the first attach to the database initiates the recovery process. When recovery is complete, the database is available to users again.

If you do not want to terminate active user processes, you have the following choices:

- `RMU Close Abort=Forcex Nocluster`—Immediately forces existing users (on this node) off the specified database and then closes the database.
- `RMU Close Abort=Forcex Cluster`—Immediately forces existing users (throughout the cluster) off the specified database and then closes the database.

In both of these cases, recovery-unit journals are recovered, no .ruj files are left in the directories, and the `RMU Backup` command works on the database. Using the `Abort=Forcex` qualifier does not guarantee immediate termination. The `Abort=Forcex` qualifier queues up a user mode asynchronous system trap (AST) that causes the executing image to run down. However, if the process is

not accepting user mode ASTs, then the image is not run down. Many factors can delay the AST for extended periods.

A database is considered closed if no one is attached to it and no previous RMU Open command was issued. If you issue an RMU Close command against a database that is not open, you receive the following error message:

```
%RDMS-F-CANTCLOSEDB, database could not be closed as requested
-RDMS-F-DBNOTACTIVE, database is not being used
%RMU-W-FATALERR, fatal error on DUA01:[DB]MF_PERSONNEL.RDB;1
```

You can also use the Path qualifier to close databases specified by the data dictionary path names. Example 4-4 shows an RMU Close command that closes the mf_personnel database and all the databases specified by the data dictionary path name CDD\$TOP.TEST.

Example 4-4 Closing Databases and Showing Users

```
$ RMU/CLOSE MF_PERSONNEL, CDD$TOP.TEST/PATH
$ RMU/SHOW USERS MF_PERSONNEL
Oracle Rdb V7.0 on node TRIxie 30-AUG-1995 13:14:33.56
  - no databases are accessed by this node
```

The only time you must close a database is when you need exclusive access to perform a maintenance operation. When you use the SQL ALTER DATABASE OPEN IS MANUAL statement with the RMU Open Access=Restricted command, access to the database is limited to users with SQL DBADM privilege for the database or OpenVMS BYPASS or SYSRV privilege (if the RMU image was not installed with the OpenVMS SYSRV privilege), so that maintenance operations can proceed without interference from other users.

4.2.1 Closing a Database and Using SQL

Close your database before entering any of the following commands and statements:

- RMU Backup (unless you use the Online qualifier)
- RMU Open
- SQL DROP

You can delete a collating sequence (SQL DROP COLLATING SEQUENCE) with the database open, but the collating sequence cannot be used by the schema or any domain in the schema.

You can delete a trigger (SQL DROP TRIGGER) with an open database with active users attached to the database and with active transactions accessing the tables for which the trigger is defined. To delete the trigger, you need only delete access to the table for which the trigger is defined.

Generally, if a particular SQL DROP statement cannot be performed with active users attached to the database and with active transactions accessing the tables or relations involved, then first close the database and in some cases open the database manually before carrying out the desired maintenance task. For a detailed list of concurrent metadata updates that are allowed and additional restrictions that may apply, see the *Oracle Rdb7 Guide to Database Design and Definition*.

If you issue the SQL DROP DATABASE statement and the SQL ALTER DATABASE DROP STORAGE AREA statement against an open database, you will receive the following error message:

```
%RDB-F-SYS_REQUEST, error from system services request
-RDMS-F-FILACCERR, error opening database root file
  AM$DISK:[HUMAN_RESOURCE1.DAT]MF_PERSONNEL.RDB;1
-SYSTEM-W-ACCONFLICT, file access conflict
```

All other metadata can be dropped or deleted with users attached to the database, but other restrictions may apply. See the *Oracle Rdb7 Guide to Database Design and Definition* and the *Oracle Rdb7 SQL Reference Manual* for additional restrictions.

If this error message is returned, issue an RMU Close command, as shown in Example 4-5, and proceed with what you were doing.

Example 4-5 Closing the Database

```
$ RMU/CLOSE MF_PERSONNEL
```

By default, the RMU Close command closes the database on a particular node (the same node on which the database was opened using an RMU Open command) by using a forced exit of all active user processes. As shown in Example 4-6, user processes are not allowed to terminate normally through attrition, and transactions in progress are automatically rolled back.

Example 4–6 Opening and Then Closing the Database

```
!  
! A user accesses the database.  
!  
$ SQL  
SQL> ATTACH 'FILENAME MF_PERSONNEL';  
SQL> !  
!  
! The DBA closes the database from another terminal on the same node.  
!  
$ RMU/CLOSE MF_PERSONNEL  
!  
! The user sees the following message on the screen:  
!  
SQL>  
%RDMS-F-OPERCLOSE, database operator requested database shutdown
```

No matter how the database was opened on a node, the RMU Close command, by default, closes the database by using a forced exit of all active user processes of the `mf_personnel` database. As shown in Example 4–6, the database shutdown message is displayed.

If the shutdown request does not close the database because it cannot terminate a particular user process, issue an RMU Show Users command to determine which process the RMU Close command could not terminate.

When a shutdown initiated by an RMU Close command with the `Noabort` qualifier is in progress, no one can attach to the database on that particular node until after the shutdown is completed, as shown in Example 4–7.

Example 4–7 Trying to Access a Closing Database

```
$ SQL  
SQL> ATTACH 'FILENAME MF_PERSONNEL';  
%SQL-F-ERRATTDEC, Error attaching to declared schema mf_personnel  
-RDB-F-SYS_REQUEST, error from system services request  
-RDMS-F-DBSHUTDOWN, database shutdown is in progress
```

To prevent users from attaching to the database after database shutdown, modify your database for manual opening when you perform maintenance operations, if you have not already done this. You may make this modification without interfering with current users. Enter the command shown in Example 4–8.

Example 4–8 Changing Database Access to Manual Open

```
$ SQL
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> OPEN IS MANUAL;
```

Then issue the RMU Close command, as shown in Example 4–9.

Example 4–9 Closing the Database

```
$ RMU/CLOSE MF_PERSONNEL
```

When you finish your maintenance operation, you can issue another SQL ALTER DATABASE statement to set the database back to automatic mode. These statements allow users to attach to the database without you or a privileged user having to issue an RMU Open command first, as shown in Example 4–10.

Example 4–10 Changing Database Access to Automatic Open

```
$ SQL
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> OPEN IS AUTOMATIC;
```

SQL statements that require a closed database before they execute need exclusive access to the database. No other users can attach to the database (no matter how you set the OPEN parameter) until you have finished your maintenance operation. However, when your database is set for automatic opening, you may need to issue several RMU Close Abort=Forcex commands before you can perform most maintenance operations, using SQL.

4.2.2 Using the Noabort Qualifier to Close a Database

The RMU Close command with the Noabort qualifier allows current user processes to terminate normally. That is, each user transaction is permitted to finish before the user process is forced off the specified database. New users are prevented from opening the database.

On OpenVMS systems, you can specify the Noabort qualifier on a specific node by using the Nocluster qualifier; or, for all nodes in the cluster, by using the Cluster qualifier.

For example, to close a database and allow users to finish their transactions for a specific node, enter the command shown in Example 4–11.

Example 4–11 Closing the Database Through Attrition for a Specific Node

```
$ RMU/CLOSE MF_PERSONNEL /NOABORT /NOCLUSTER
```

In Example 4–11, users already attached to the database on this node remain attached until their transactions finish and they automatically detach.

When the last user finishes his transaction and automatically detaches, the database is closed. Use the RMU Show Users command to see if active users are attached to the database, as shown in Example 4–12.

Example 4–12 Showing Active Users Attached to the Database

```
$ RMU/SHOW USERS MF_PERSONNEL
Oracle Rdb V7.0 on node TRIxie 30-AUG-1995 13:34:22.46
database AM$DISK:[RESOURCE1.DAT]MF_PERSONNEL.RDB;1
  * local database shutdown is in progress
  - global buffer count is 50
  - maximum global buffer count per user is 2
  - 48 global buffers free
  - 1 active database user
  - 20C01F2F:0 - orion, orion - active user
    - image SYS$SYSTEM:SQL$.EXE;1
    - 2 global buffers allocated
```

Example 4–12 does not show a forced shutdown, but it is an example of a closing of the database through attrition because of the Noabort qualifier. When all user transactions have completed, you see the output shown in Example 4–13.

Example 4–13 Showing No Active Users Attached to the Database

```
$ RMU/SHOW USERS MF_PERSONNEL
Oracle Rdb V7.0 on node TRIxie 30-AUG-1995 13:35:34.56
  - no databases are accessed by this node
```

The RMU Close command with the Noabort and Nocluster qualifiers affects only the node on which it is issued. In a cluster environment, a user on another node can still attach to the database even after you issue an RMU Close command with the Noabort qualifier on your node. This would prevent you from performing most maintenance functions with SQL. However, you can use the Cluster qualifier with the RMU Close command to close a database across a cluster.

By including the Cluster and Noabort qualifiers on the RMU Close command, you can close the database on all nodes of your cluster accessed by that database, as shown in Example 4–14.

Example 4–14 Closing the Database Through Attrition for All Nodes

```
$ RMU/CLOSE MF_PERSONNEL /NOABORT /CLUSTER
```

To shut down your database on only some of the nodes of a cluster, you must issue an RMU Close command with the Noabort qualifier, specifying the Nocluster qualifier from each node you want to close. To see if there are users on any nodes, use the RMU Show Users command. See the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information. ♦

See the *Oracle RMU Reference Manual* for more information on the syntax of the RMU Open and RMU Close commands.

Verifying the Integrity of Your Oracle Rdb Database

On rare occasions, media failure, hardware failure, or software error can corrupt your database. In a worst case situation, if the corruption is limited to a portion of your database that is seldom accessed, you might not receive an immediate access violation and bugcheck dump. Meanwhile, you may have performed other regularly scheduled maintenance activities such as full and incremental backup operations without realizing that the corruption was there. For example, if the problem is not detected by the checksum check in a backup operation, then your backup file also contains the corruption.

This kind of problem can go unnoticed for some time, but at some point it can produce an access violation and bugcheck dump. You then begin the process of determining when the problem first occurred, what caused it, and most important, how you can get up and running again with a good version of your database. If you are not able to restore and recover your database because all backup files contain the corruption, you must rebuild the database from a set of flat files.

It is important to verify your database regularly. Scheduled, full verify operations detect problems and allow you to take prompt action.

5.1 Why You Should Verify Your Database

Verifying your database is an essential and integral part of the set of maintenance tasks that you need to perform. You should verify the integrity of your database at the following times:

- Before backup and optionally after restore operations
You should verify your database prior to backing it up. Doing this ensures that database corruption is detected before the backup operation is attempted. You can optionally verify your database after you have restored it, depending upon whether you suspect a problem exists. For example, if there is a disk problem or a tape reading problem, the restore operation may eventually succeed but display tape read errors or disk write errors.

If you suspect a problem, verify your database before proceeding further. See Section 5.10 for more information on performing verify operations to troubleshoot suspected problems.

- On a regular basis

You should verify your database by performing a full verification using the RMU Verify All command on a regular schedule based on available time, the size of the database, and resource constraints. More information on devising such a plan is presented in Section 5.6.

- If you suspect a problem exists

On rare occasions, a situation may occur in which one or more error messages display that indicate a problem with your database. You might get an access violation and a bugcheck dump, or you may receive one or more error messages, depending on what you were doing before the message was received. The bugcheck dump usually includes dumps of database pages, indexes, and so forth. You can verify these pages and indexes to determine if pages are corrupt. Perform an RMU verify operation to troubleshoot the suspected problem. See Section 5.10 for more information on performing verify operations to troubleshoot suspected problems.

- Following a system failure

You should always perform a full verification of your database if a system failure has occurred. Oracle Rdb is designed to prevent corruption from occurring under these circumstances. For example, Oracle Rdb uses recovery-unit journal (.ruj) files to recover incomplete (uncommitted) transactions on the database when user processes abort, nodes go down in a cluster environment, or there is a complete system failure. As a precaution, however, Oracle Rdb recommends that you perform a verify operation of your database following a system failure.

5.2 Causes of Database Corruption

Database corruption usually results from one of the following causes:

- Abnormal image termination during a batch-update transaction
- Hardware errors such as the flipping of bits
- A software error due to a system failure

When you access a database in batch-update mode, a recovery-unit journal is not maintained for your transaction. Corruption can occur due to any failure such as a hardware problem, a software error, abnormal image termination, or a verb failure. An update verb that generates any type of error, such as violating a constraint definition, causes an automatic Oracle Rdb software rollback, corrupting the storage area. You must manually restore your database to the state it was in before you began your batch-update transaction. Corruption can also result from exception to a ROLLBACK statement.

Corruption caused by system failure can go unnoticed for days or weeks. The RMU Backup command checks for certain types of corruption and displays a message if it finds corruption. However, the only way to ensure finding most corruption problems is to perform a verify operation.

Under certain circumstances, such as accessing a database in batch-update mode and getting a verb failure, database corruption can go undetected if a database verification policy is not in place. Backup and journal files might also contain corrupt database pages.

5.3 What Happens When You Verify Your Database

By default, the RMU Verify command starts a read/write . . . protected read transaction against the entire database. If a user has already started a read/write . . . exclusive write or batch-update transaction, an AREABUSY error occurs when you issue the RMU Verify command. If users have started shared, protected, or read-only transactions, you can use the RMU Verify command. While you are verifying the database, subsequent users can start read-only or read/write . . . read transactions. A user cannot start a read/write . . . write transaction until the verify operation is complete.

The database must be open to do a verify operation. You can open the database for restricted access by closing the database and then opening it again using the RMU Open command with the Access=Restricted qualifier. Only users with SQL DBADM privilege can access the database during a verify operation. If necessary, you can open the database with unrestricted access to allow read-only transactions during the verify operation. The online verify operation appears as a very I/O-intensive user process to other database users.

For both single-file and multfile databases, you can also verify a closed database that is opened using the OPEN IS MANUAL clause of an SQL ALTER DATABASE statement. However, users with SQL DBADM privilege can access the database during the verify operation. If you want to restrict access to the database for all users because, for example, database corruption is suspected and you do not want any further damage to occur, you can either notify users that the database is temporarily unavailable or close the database

and open it for restricted access, start a verify operation, and then close the database in an orderly fashion by using the `Noabort` and `Cluster` qualifier on the `RMU Close` command. This command starts an orderly shutdown throughout the cluster, does not force users off the database, prevents new users from attaching to the database, and permits the verify operation to continue until it is complete. See Chapter 4 and the *Oracle RMU Reference Manual* for more information on the `RMU Close` command.

The `RMU Verify` command stores timestamps in the database root file (`.rdb`) that indicate the time of the last full or incremental verify operation. The *Oracle RMU Reference Manual* describes the `RMU` privileges required to run the `RMU Verify` command and to update the root file with timestamp information. The `RMU Verify` command compares the page timestamps and the verify timestamps when performing an incremental verify operation. This operation checks only those database pages that have changed since the last full or incremental verify operation, but it must fetch all pages to make this comparison.

If the verify operation detects a problem, an error message is returned that tells you the page number on which the corruption occurred. Depending upon what part of the verify operation detected the corruption, more detailed information is returned that describes the corruption. Each error message is described in more detail in online Help under the category `RMU_ERRORS`.

If a part of your database becomes corrupt due to a media failure, hardware failure, or software error, and a user accesses that part of the database between verify operations, the database corruption may produce an error message or a bugcheck dump. Should this happen, you may still be able to restore your database from an uncorrupt backup file and roll it forward. See Appendix A for more information on handling bugcheck dumps.

5.4 What the Full Verify Operation Checks

Issuing the `verify` command causes `RMU` to check the internal data structures of the database, confirming that they are not corrupt or providing a warning message that there is a problem. The following tasks are performed during a full verify operation (`RMU Verify All`):

- A check of the `.rdb` file
 - Checks the system of pointers from the `.rdb` file to the other database files (storage area (`.rda`), snapshot (`.snp`), and after-image journal (`.aij`) files).
 - Checks that each of these database files corresponds with the information in the `.rdb` file with respect to file name, file type, and version number.
- A check of all constraints to ensure that the data integrity is intact

- A check of all logical areas

Checks the page verification of all database pages, which includes verifying the page header, the page checksum, the page line indexes, and the page tail.

- The page header is checked to determine if the page is indeed what it says it is. That is, if the page is supposed to be page 25, it is in fact page 25.
- A page checksum check involves determining if the total value of all bytes on the page equals the checksum value stored on the page. During normal database activity, if information changes on the page, the checksum for the page is recalculated and updated. If some problem has caused information to be written to the page abnormally, the page becomes corrupt. Therefore, if the verify operation checksum check finds a problem, and the page checksum is different from the calculated checksum, the page is corrupt. If the checksum values agree, it is likely that the database page is not corrupt. It is possible that a checksum can be computed and stored correctly on the page but that the corruption is not detected before the page is written to disk. Therefore, the page could be corrupt and not detected by the checksum-only verify operation.
- A page line index check involves checking to see that all line entries for storage segments correspond to a line index entry, and that the available space on the page is exactly the amount stated. If the line index check detects an incorrect number of line index entries, a missing line entry, an extra line entry (storage segment), incorrect free space, or free space where it should not be, then the page is corrupt. Such problems can also be shown in a display of the page by the appearance of the words “junk” and “overlap.”

During a line index check, all lines on the page are fetched and verified. If any one of these lines contains data records, then list or segmented string verification occurs. List data is checked in both read/write and read-only storage areas on read/write disk devices, and in write-once storage areas on write-once, read-many (WORM) optical disk devices. This check determines that:

- + Each (data) segment database key (dbkey) has the correct logical area database ID (dbid) in it
- + Each (data) segment can be fetched
- + The total length of the list is correct
- + The total number of (data) segments in the list is correct

- + The length of the longest (data) segment in the list is correct
- Lists in write-once storage areas on WORM optical disk devices are also checked to be sure that:
- + Each pointer segment dbkey has the correct logical area dbid in it
 - + Each pointer segment can be fetched
 - + The length of each data segment is correct
 - + The number of data segments in each pointer segment is correct
 - + The total number of pointer segments in the list is correct
- The page tail check involves checking to see that the pointer points to the correct logical area for a uniform page format storage area.
 - A check of all storage pages

Checks all area inventory pages (AIP), area bit map (ABM), space area management (SPAM), and data pages. For example, with ABM page verification, the ABM bit vector is checked and for any unusual SPAM pages it finds, it checks the logical areas listed in those SPAM pages to verify that there is actually a clump of pages that belongs to the same logical area as the ABM page.
 - A check of all index structures

Checks that all index structures for all sorted and hashed indexes exist and that data records can be fetched.

For sorted indexes, a check is made to determine if the keys are sorted. In addition, during an index verification, data records are fetched and verified if the Data qualifier is specified.
 - A check of all storage areas

Checks all pages in the list of storage areas specified and performs a page verification check of these pages. However, if page verification has already been done for a page prior to this time, for example, from an index or logical area check, the page is not verified again. It also checks and updates the page corruption table for each storage area.
 - A check of all snapshot areas

Checks all snapshot areas in the list of storage areas specified in the Areas qualifier. The snapshot area is only verified to the level of the page header, which includes checking the number of the storage area, the page number, and the page checksum.
 - A check of all external functions

Checks all external functions and verifies that the shareable image exists on the system, is located where expected, is accessible, and that all entry points are correct.

5.5 What Problems the Full Verify Operation Can Detect

The verify operation detects missing database files, wrong versions of database files, and other inconsistencies between the .rdb file and the files that comprise the database. It also detects missing or incomplete AIP, ABM, SPAM, and data pages. The verify operation detects missing or incorrect page header information; incorrect page checksums; any inconsistencies among the page line indexes for the page; any missing, partial, or extra storage segments; incorrect free space for the page; and missing or incorrect page tail information. In addition, duplicate table names are detected within storage areas that might result from corruption of a database.

Any missing index structures are detected, such as missing B-tree nodes, missing duplicate B-tree nodes, missing B-tree leaf nodes, and missing data records; for hash index structures, any missing system records, missing hash buckets, missing hash bucket entries, missing duplicate node records, missing data records, and missing list data are detected. Any of these structures could be missing because the pointer to it was corrupt.

Snapshot (.snp) file page headers are checked for incorrect or missing storage area numbers, page numbers, and checksums.

List data is checked in both read/write and read-only storage areas on read/write disk devices and in write-once storage areas on WORM optical disk devices for an incorrect logical area dbid for each (data) segment dbkey and to make certain that each (data) segment can be fetched, for incorrect total lengths of the list, for incorrect total number of (data) segments in the list, and for incorrect lengths of the longest (data) segment in the list. In addition, lists in write-once storage areas on WORM optical disk devices are checked for an incorrect logical area dbid for each pointer segment dbkey and to make certain that each pointer segment can be fetched, for incorrect lengths of each data segment, for incorrect numbers of data segments in each pointer segment, and for incorrect total numbers of pointer segments in the list.

Data integrity problems are detected, such as missing data, invalid data, or misplaced data as detected from the constraint check. All defined constraints in the database are loaded, executed, and checked. The constraint check does not attempt to determine whether data within rows is reasonable or plausible.

Page checksum problems are checked against page entries in the page corruption table. If a page is found to have an incorrect checksum value and is not found in the page corruption table, the page corruption table is updated. Page corruption table updates are made during verify operations that involve verification of storage areas.

Finally, any problems with external functions are detected; for example, are the shareable images present on the system where they are expected, are they accessible, and are all the entry points correct?

When a problem is detected using the verify operation, specific error messages are returned to indicate the nature of the problem. In cases when a metadata problem exists, such as a metadata corruption error condition, specific information is returned to the user. In cases where more than one storage area is found to be corrupt, the corrupt storage areas are listed. In the case of a corrupt B-tree index, the path to the corrupt portion of the B-tree index is provided from the root node.

Note

Verification of a piece of data involves comparing two values. If the comparison does not succeed, either of the two values could be corrupt. Therefore, read the verify error messages carefully. Additional information describing each error message can be found in online Help under the category `RMU_ERRORS`.

When a problem is detected and the database is found to be corrupt, the exact problem and its location is indicated. Depending upon the corruption problem, the database may need to be restored from backup files and recovered to its most current state by applying the .aij files if after-image journaling is enabled. If the problem is isolated to a storage area, then just that storage area must be restored and recovered. If the problem is isolated to a page, then just that page must be restored and recovered. If the problem is isolated to an index structure, then the index must be deleted and rebuilt. More information on devising a strategy to pinpoint and solve these problems can be found in Section 5.6.

5.6 Devising a Full Verify Strategy to Detect Problems

Oracle Rdb recommends that you perform a full verify operation regularly, including a fetch of the rows as part of the index check. As a trade-off of thoroughness versus performance, you can exclude the row fetch check that is part of the index check, but this verify operation should not be substituted for a full verify operation that includes the row fetch check. The two verify operations are recommended in the following order of use:

1. A full verify operation of your database (including a row fetch check) to detect problems prior to each backup operation

You should perform the fullest possible verify operation prior to a database backup operation. A full verify operation is performed by entering the RMU Verify command and specifying the All qualifier. By default, all paths to rows are checked and the rows fetched as part of index verification. This is the most thorough verify operation that you can make of your database and consequently demands the most of your CPU and system configuration. Every page in the database is checked. More information on using this command is presented in Section 5.7.

2. A full verify operation of your database (excluding a row fetch check) whenever you suspect problems exist

To troubleshoot problems, you can perform a full verify operation of your database that excludes the row fetch check. In this case, use the RMU Verify command with the following qualifiers: Root, Constraint, Indexes, Nodata, Areas=*, Snapshots, Lareas=*, Segmented_Strings. This command does as thorough a check of your database as the RMU Verify All command but excludes fetching the rows when index verification is performed. It is less demanding of your disks because fewer I/O operations are required. This is a useful verify operation. For example, if you suspect a problem exists and you want better performance from the verify operation. See Section 5.7 for more information on using this command.

Considering your CPU size and your system configuration, there is a trade-off between the thoroughness of the verify operation and the demand imposed on your system in performing a full verify operation (RMU Verify All). The ideal choice is to have the time and resources to perform this operation regularly. In reality, it may not always be possible to do a full verify operation of your database due to the size of your database, the amount of round-the-clock database update activity, limited resources, and so forth. The following verify method is not as thorough and is not a substitute for a regular full verify operation that includes a row fetch check as part of index verification.

Oracle Corporation recommends the following variation of a full verify operation as a practical alternative when resources are limited:

- Split the full verify operation (RMU Verify All) by storage area.
 - Prepare a list of indexes and logical areas in each storage area by using the RMU Analyze command.
 - Run a separate full verify operation for each storage area by specifying the list of indexes and logical areas in each storage area.

The elapsed time required to verify each storage area. Devise a schedule based on your time constraints to accomplish a full verify operation, using the time available each day, until all storage areas are fully verified. Elapsed times are relative to your system load, but they are still useful for planning purposes when verify operations are performed regularly and your system loads are comparable.

A full verify operation of your database is accomplished as if the verify operation were run and completed during a single point in time. That is, breaking the task into subtasks and running subtasks on consecutive days as time is available produces the same final result—a full verify operation of your database. For example, if you know that a full verify operation of all storage areas requires a total of 18 hours, you can plan to accomplish the task in approximately 6 subtasks of more or less than 3 hours each over a period of 6 days, if the areas are of similar size and complexity.

Other variations of the full verification procedure include the following:

- Combinations of full and incremental verify operations performed according to a schedule

Based on estimates of elapsed time required to perform a full verify operation on the database, you can schedule incremental verify operations between the full verify operations. Incremental verify operations can be done daily or according to the update activity on the database, and full verify operations can precede full backup operations of the database.

- A continuous verify operation, using the `Transaction_Type=Read_Only` qualifier on the RMU Verify command

You can run the verify operation continuously as a read-only transaction. This alternative is useful when little time is available to verify the database because users are always updating the database.

You can use the RMU Extract command and specify the Item=Verify qualifier to create a command procedure containing a script of partial RMU Verify commands or verify command partitions that are equivalent to a full verification (RMU Verify All) of the database. You can submit these partial command procedures to different queues to do a full verify operation in parallel, or you can spread out a full verify operation over several days by verifying a piece of the database at a time.

A partitioning algorithm makes the following considerations when composing the RMU Verify command into its partial scripts (command partition):

1. Each storage area is assigned to a partition.
2. For each table in the database, if the table is not partitioned, the table is put in the partition corresponding to that storage area; otherwise, if the table is partitioned across several storage areas, the partitions corresponding to all of the storage areas are merged into one partition, and the table is added to this new partition.
3. For each index in the database, the same process as shown in step 2 is followed.

The scripts of partial RMU Verify commands are output in the form of a command procedure. Each partial command or command partition is preceded by a label of the form stream_ *n*: where *n* is an integer greater than or equal to 1. To execute the command at label stream_3:, you invoke your command procedure by using the following formats:

- On OpenVMS systems, use the following syntax:

```
$ @command-procedure-name stream_3
```

- On Digital UNIX systems, use the following syntax:

```
$ sh script-name stream_3
```

The resulting command procedure accepts up to four parameters, P1–P4, as shown in Table 5–1.

Table 5–1 Parameters for the Generated Command File

Parameter	Option	Description
P1	stream_n	Identifies the stream to be executed, where <i>n</i> is the “number” of the RMU Verify stream to be executed. If omitted, all streams will be executed.
P2	[No]Log	Indicates whether to use the Log qualifier in the RMU Verify command line. If omitted, the DCL verify switch value is used.
P3	Read_Only Protected Exclusive	Supplies the RMU Verify Transaction_Type value. If omitted, Transaction_Type = Protected is used.
P4		Provides the name of the output file for the RMU Verify Output qualifier. If omitted, Output is sent to the standard output device. (SYSSOUTPUT on OpenVMS systems and stdout on Digital UNIX systems).

See the *Oracle RMU Reference Manual* for more information and an example of running the RMU Extract Items=Verify for the mf_personnel sample database.

5.7 Interaction of RMU Verify Command Qualifiers

This section describes other available RMU Verify command qualifiers and the specific parts of your database they check. There is a trade-off between the performance of specific verify operations and their thoroughness. The size of your CPU, your system configuration, the size of your database, and the nature of your application all determine when and what verify operations you choose for troubleshooting suspected problems in the most efficient manner.

To devise a strategy for using specific verify operations to isolate a problem in your database, you must understand what each RMU Verify qualifier does and how each qualifier works in combination with other qualifiers. Possible combinations of RMU Verify qualifiers are listed here with a brief summary of what they do.

- **RMU Verify (implies Root)**
Performs by default only an .rdb file verification and a full page verification of all AIP pages in the default system storage area and all ABM pages in each uniform storage area. By default, none of the following checks is performed: other storage areas besides the default system storage area, .snp files, logical areas, indexes, constraints, and lists. Specify either the

All qualifier or other specific qualifiers if you want the verify operation to perform other checks.

- **RMU Verify All Incremental**

Performs a verify operation of only the database pages that have changed since the last full or incremental verify operation, based on timestamps stored in the .rdb file and on pages. If the Incremental qualifier is not specified, the timestamp on the page is ignored and any page can be verified, depending on what other qualifiers are specified.

Note

Oracle Rdb recommends that you use the Incremental qualifier only with the All qualifier for the RMU Verify command. Use of the Incremental qualifier with other combinations of qualifiers performs verify operations that are not intuitively clear. The use of timestamps for the Incremental qualifier is given here to clarify the behavior expected when using the Incremental qualifier with other combinations of RMU Verify command qualifiers.

The timestamp on a database page is updated whenever the page is updated. The timestamps in the .rdb file for full and incremental verify operations are updated only if the All qualifier is specified. Therefore, two successive incremental verify operations of the same storage area of the database perform the same verifications if the All qualifier is not specified. In particular, the second incremental verify operation does not omit pages verified by the first incremental verify operation. This is currently a restriction.

All pages are fetched to check the timestamp and to determine if the page needs to be verified. The Incremental qualifier is useful if your CPU is constrained or in heavy use due to other jobs running or the submission of a large number of batch jobs that run at night at the same time you run a verify operation. Under these circumstances, when you do an incremental verify operation the elapsed time of verification may be less.

- **RMU Verify Areas=* Checksum_Only**

Performs only a checksum verification of pages (excluding page header and page line index verification) for the list of storage areas specified in the Areas qualifier, and checks and updates the page corruption table. This reduces the degree of page verification and provides better performance even though it requires fetching all the pages to check the checksum of the page. This command is useful to troubleshoot problems and concentrate

further verify operations to specific storage areas or logical areas. For example, if you find a problem with a certain page with the checksum-only verification, then that page can be verified completely by using other qualifiers such as the Index or Larea qualifiers. The Checksum_Only qualifier must be specified with the Areas qualifier to work.

- **RMU Verify All**

Performs a verification equivalent to entering, in the following order, the Constraints, Root, Indexes, Data, Areas, Snapshots, Lareas, Segmented_Strings, and Functions qualifiers. If you do not specify the All qualifier, the default is the Root qualifier. Oracle Corporation recommends that you use the Incremental qualifier only with the All qualifier.

- **RMU Verify Root**

Verifies the pointers to the database files (.rda, .snp, and .ajj) for multfile databases. This command also verifies that each database file corresponds with what is stated in the .rdb file and belongs to the database, and checks file names, file types, and file version numbers. AIP pages and all ABM pages in each uniform storage area are verified. If you specify the Noroot qualifier without qualifiers, only the AIP pages are verified. If you specify the Noroot qualifier and the Areas or Lareas qualifier, ABM and SPAM pages are verified when other pages in the storage area or logical area are verified.

- **RMU Verify Areas**

Verifies the storage areas specified and performs page-level verification of all pages in the storage area, and checks and updates the page corruption table. The elapsed time for area verification is indicated in the log messages. This is useful information when you do regular verification of the database and have saved the verify logs. If a corruption problem is detected and you want to verify a small part of the database, you can look at the previous logs to determine how long it might take to verify a specific storage area and decide on what verification to perform.

- **RMU Verify Areas=* Snapshots**

Verifies the snapshot area and the storage area from the list of areas specified in the Areas qualifier, and checks and updates the page corruption table. The snapshot area is verified only for the page header level, which includes checking the number of the storage area, the page number, and the page checksum. When you specify the All qualifier, Oracle RMU automatically includes verification of all snapshot areas. You can use the Snapshots qualifier only with the Areas qualifier.

- **RMU Verify Larea=***

Verifies the logical areas specified, and performs page-level verification of all pages associated with that logical area. For uniform storage areas, only those clumps of pages associated with the logical areas within the storage area are checked. For mixed storage areas, page clumps are absent, so a logical area can be placed on any combination of pages. Therefore, all pages in the storage area are checked.

There is little difference in the elapsed time to perform either an `RMU Verify Area=*` command or an `RMU Verify Area=* Larea=*` command.

- **RMU Verify Larea=*Segmented_Strings**
Verifies list data for all columns in all tables and for all items that are verified when the Larea qualifier is specified. When only logical area verification is considered, segmented string verification takes place as each data record from each page of a logical area is fetched and verified.
- **RMU Verify Constraints**
Verifies all constraints by loading and executing each one to check the data integrity, including references to external functions and their entry points.
- **RMU Verify Indexes**
Verifies the integrity of all indexes or any indexes specified by doing a more complete index structure check for both sorted and hashed indexes, and fetches the data records. This command also performs verification of hashed indexes within the same area in one pass. That is, when more than one hashed index occurs in the same storage area, all such indexes are verified in one pass through the storage area, rather than in multiple passes (one for each hashed index).
- **RMU Verify Indexes [No]Data**
Verifies the structure of all indexes or the indexes specified. The Nodata qualifier allows for a faster but less thorough check of the index by not checking that data records can be fetched. The default is the Data qualifier. Using the Nodata qualifier can be a significant performance improvement for verifying sorted indexes, but not for hashed indexes.

Section 5.10 describes how to use these qualifiers to troubleshoot suspected problems.

5.8 Measuring and Improving Verification Performance

When you perform a verify operation, the following information can be useful to monitor performance of the verify operations and to devise verify operation strategies:

- If you are using an `RMU Verify Indexes=(sorted-index-list) Data` command to obtain a list of only sorted indexes, improve the performance of the operation by increasing the number of database buffers with the `SQL ALTER SCHEMA` statement and specify a greater number of buffers with the `NUMBER OF BUFFERS IS` clause. Because more pages are in the buffer and fewer pages are flushed from the buffer as the sorted index is scanned and rows are fetched, performance is improved. However, increasing the number of buffers may increase page contention because the verify operation will not give up a page until that page is flushed from the buffer pool.
- To improve verification performance when using either the `All` or the `Larea Segmented_Strings` qualifier to verify segmented strings, increase the number of database buffers. When a database page contains segmented string segments from more than one segmented string, having more buffers increases the chance that the page will be in memory when the second and subsequent segmented strings on the same page are verified.
- Use the `RMU Verify Log` and `Output` qualifiers. These qualifiers record a log of all verify operations performed on your database. Save all your verification logs to maintain a verification history of your database for future reference. This information is useful for devising a schedule of verify operations or modifying your verify schedule based on elapsed times required to complete an operation such as verifying each storage area, devising a troubleshooting plan based on the time available to track down a problem, or determining when a problem first appeared. Elapsed times for each storage area are comparable only for the same command and set of qualifiers specified. Elapsed times are also affected by the load on your system when verify operations are performed and can increase as the amount of data in the database increases. Therefore, elapsed times are comparable only when verify operations are performed under identical system loads.
- On OpenVMS systems, use the `DCL MONITOR` command to determine if there is a CPU bottleneck while the verify operation is running.

- On OpenVMS systems, use the DCL SHOW PROCESS/MEMORY command to monitor memory usage for the process running the verify operation on a specific CPU. If you are verifying sorted indexes, you can increase the number of database buffers to improve the verification performance.
- Use the RMU Show Statistics command to monitor data file read operations and I/O stalls to check for disk bottlenecks while the verify operation is running. See the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information on using the RMU Show Statistics command.
- Use the RMU Dump command to display or print specific database pages for further inspection when you detect problems such as page checksum problems.

5.9 Examples of Verify Operations on the mf_personnel Sample Database

A full verify operation (RMU Verify All) on the mf_personnel sample database is the most thorough verification possible. Example 5–1 shows portions of the log file for a full verify operation.

Example 5–1 The Log File from a Full Verify Operation

```
$ RMU/VERIFY/ALL /LOG /OUTPUT=MFPERS_FULL_VERIFY.LIS -
_ $ DB_DISK:[MFPERS]MF_PERSONNEL
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-BGNVCONST, beginning verification of constraints for database
DB_DISK:[MFPERS]MF_PERSONNEL.RDB;1"
%RMU-I-ENDVCONST, completed verification of constraints for database
DB_DISK:[MFPERS]MF_PERSONNEL.RDB;1"
%RMU-I-DBBOUND, bound to database "DB_DISK:[MFPERS]MF_PERSONNEL.RDB;1"
%RMU-I-OPENAREA, opened storage area RDB$SYSTEM for protected retrieval
%RMU-I-BGNAIPVER, beginning AIP pages verification
%RMU-I-ENDAIPVER, completed AIP pages verification
```

(continued on next page)

Example 5-1 (Cont.) The Log File from a Full Verify Operation

```
%RMU-I-OPENAREA, opened storage area EMPIDS_OVER for protected retrieval
%RMU-I-OPENAREA, opened storage area EMPIDS_MID for protected retrieval
%RMU-I-OPENAREA, opened storage area EMPIDS_LOW for protected retrieval
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-I-ENDABMSPM, completed ABM pages verification
%RMU-I-BGNNDXVER, beginning verification of index COLL_COLLEGE_CODE
%RMU-I-OPENAREA, opened storage area EMP_INFO for protected retrieval
%RMU-I-ENDNDXVER, completed verification of index COLL_COLLEGE_CODE
.
.
.
%RMU-I-OPENAREA, opened storage area SALARY_HISTORY for protected retrieval
%RMU-I-ENDNDXVER, completed verification of index SH_EMPLOYEE_ID
%RMU-I-BGNHDXVER, beginning verification of hash index EMPLOYEES_HASH
as part of EMPIDS_LOW storage area
%RMU-I-BGNHDXVER, beginning verification of hash index JOB_HISTORY_HASH
as part of EMPIDS_LOW storage area
%RMU-I-ENDHDXVER, completed verification of hash index EMPLOYEES_HASH
as part of EMPIDS_LOW storage area
%RMU-I-ENDHDXVER, completed verification of hash index JOB_HISTORY_HASH
as part of EMPIDS_LOW storage area
.
.
.
%RMU-I-BGNSEGPAG, beginning verification of RDB$SYSTEM storage area
%RMU-I-ENDSEGPAG, completed verification of RDB$SYSTEM storage area
elapsed time : 00:00:00.71
%RMU-I-OPNSNPARE, opened snapshot area RDB$SYSTEM for protected retrieval
%RMU-I-BGNSNPVER, beginning snapshot verification of RDB$SYSTEM storage area
%RMU-I-ENDSNPVER, completed snapshot verification of RDB$SYSTEM storage area
elapsed time : 00:00:01.66
%RMU-I-BGNSEGPAG, beginning verification of EMPIDS_LOW storage area
%RMU-I-ENDSEGPAG, completed verification of EMPIDS_LOW storage area
elapsed time : 00:00:00.00
%RMU-I-OPNSNPARE, opened snapshot area EMPIDS_LOW for protected retrieval
%RMU-I-BGNSNPVER, beginning snapshot verification of EMPIDS_LOW storage area
%RMU-I-ENDSNPVER, completed snapshot verification of EMPIDS_LOW storage area
elapsed time : 00:00:00.11
.
.
.
```

(continued on next page)

Example 5–1 (Cont.) The Log File from a Full Verify Operation

```
%RMU-I-OPENAREA, opened storage area RESUMES for protected retrieval
%RMU-I-BGNSEGPAG, beginning verification of RESUMES storage area
%RMU-I-ENDESEGPAG, completed verification of RESUMES storage area
                    elapsed time : 00:00:01.90
%RMU-I-OPNSNPARE, opened snapshot area RESUMES for protected retrieval
%RMU-I-BGNSNPVER, beginning snapshot verification of RESUMES storage area
%RMU-I-ENDSNPVER, completed snapshot verification of RESUMES storage area
                    elapsed time : 00:00:00.78
%RMU-I-BSGPGLARE, beginning verification of RDB$RELATIONS logical area
                    as part of RDB$SYSTEM storage area
%RMU-I-ESGPGLARE, completed verification of RDB$RELATIONS logical area
                    as part of RDB$SYSTEM storage area
.
.
.
%RMU-I-BSGPGLARE, beginning verification of RESUMES logical area
                    as part of RESUMES storage area
%RMU-I-ESGPGLARE, completed verification of RESUMES logical area
                    as part of RESUMES storage area
%RMU-I-CLOSAREAS, releasing protected retrieval lock on all storage areas
%RMU-I-BGNEXTFNC, beginning verification of external functions.
%RMU-I-ENDEXTFNC, completed verification of external functions.
%RMU-S-ENDVERIFY, elapsed time for verification :    0 00:03:07.68
```

In Example 5–1, a verification is performed that is equivalent to entering, in the following order, the Root, Constraints, Indexes, Data, Areas, Snapshots, Larea, Segmented_Strings, and Functions qualifiers. That is, the .rdb file is checked first, followed by a check of all constraints, a check of all indexes and a fetch of the rows, a check of each storage area and related snapshot area, a check of all logical areas, and finally a check of all external functions, if any exist.

By default, the RMU Verify command checks only the .rdb file and opens the system storage area to perform a page verification. If the RMU Verify command detects any problems, it displays a message that indicates the nature of each problem. Enter the command shown in Example 5–2 to perform a default verify operation on the mf_personnel database (use the Log qualifier to display each step).

Example 5–2 Verifying the Integrity of the Database

```
$ RMU/VERIFY /LOG DB_DISK:[MFPERS]MF_PERSONNEL
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-DBBOUND, bound to database "DB_DISK:[MFPERS]MF_PERSONNEL.RDB;1"
%RMU-I-OPENAREA, opened storage area RDB$SYSTEM for protected retrieval
%RMU-I-BGNAIPVER, beginning AIP pages verification
%RMU-I-ENDAIPVER, completed AIP pages verification
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-I-ENDABMSPM, completed ABM pages verification
%RMU-I-CLOSAREAS, releasing protected retrieval lock on all storage areas
%RMU-S-ENDVERIFY, elapsed time for verification : 0 00:00:16.17
$
```

In Example 5–2, the RMU Verify command checked the .rdb file in the mf_personnel database. No database corruption was detected.

After you recover from a system failure, use the RMU Verify command with the Areas and Checksum_Only qualifiers to determine if the database was corrupted as a result of the failure. If you are checking the integrity of the database after a system failure, you can use the Transaction_Type qualifier to prevent anyone else from accessing the database until you are certain it was not corrupted. For example, specify the Transaction_Type=Exclusive qualifier to prevent all access to the database during verification, as shown in Example 5–3.

Example 5–3 Using Exclusive Access During a Verify Operation

```
$ RMU/VERIFY/AREAS=*/CHECKSUM_ONLY/TRANSACTION_TYPE=EXCLUSIVE -
_ $ /LOG /OUTPUT=MFPERS.VFY_LOG DB_DISK:[MFPERS]MF_PERSONNEL
```

The Transaction_Type qualifier sets the area lock for the storage areas being verified. The valid options are: exclusive, protected, and read-only. The protected mode is the default. All transaction types use read as the only valid access mode. Thus, by default, the RMU Verify command starts a read/write reserving table name list for protected read transaction.

5.10 Troubleshooting Suspected Problems

Troubleshooting involves trying to isolate and detect a problem that appeared as an error message returned to a user, an exception, a bugcheck dump, or the result of a specific verify operation. Incomplete disk updates (only a fraction of a page is written to disk) might be caused by a disk problem, a system (software or hardware) failure, or a power failure. Troubleshooting these problems involves a trade-off between the performance of the verify operation and its thoroughness.

If you perform routine full verify operations on your database and one or more error messages display, it may be necessary to perform a more thorough verification of a specific part of the database. For example, with an index problem, you could use the Data qualifier to fetch all records to try to pinpoint the problem. The performance of this check may be slower because all index records and all data pages pointed to by the index must be fetched and checked.

If you suspect a problem exists, such as a disk or system problem, you might perform a fairly quick and specific verification of your entire database, such as a check of each page checksum. If one or more pages are found to be corrupt, the corrupt page table (CPT) is updated to indicate the pages in each storage area that are corrupt. With page checksum errors, the interesting factors are the location of the problem spots in the database and the frequency with which they occur. Clustered errors or a high frequency of errors generally indicate disk or system maintenance problems. To correct these problems, you fix the cause of the problem and restore and recover your database from uncorrupted backup and .aij files.

The CPT logs known corrupt or inconsistent database pages. You can display the contents of the CPT by using the `RMU Show Corrupt_Pages` command. If no corrupt pages are logged in the CPT, the content includes a message indicating that the CPT is empty.

Pages marked as corrupt from a verify operation are logged to the CPT display, as shown in Example 5-4. Be sure to verify your database prior to displaying the CPT to obtain the most current information. The CPT can hold 127 entries and adds 4K bytes to the root file. If there are more than 127 corrupt pages in the database, the area with the most entries is marked as corrupt and its entries removed to allow space for more entries to be logged in the CPT.

Example 5–4 Display of the Corrupt Page Table

```
$ RMU/SHOW CORRUPT_PAGES MF_PERSONNEL
*-----
* Oracle Rdb V7.0-00                               20-OCT-1995 16:16:46.51
*
* Dump of Corrupt Page Table
*   Database: DB_DISK:[MFPERS]MF_PERSONNEL.RDB;1
*
*-----
Entries for storage area EMPIDS_LOW
-----
Page 2
- AIJ recovery sequence number is -1
- Area ID number is 2
- Consistency transaction sequence number is 0
- State of page is: corrupt

Page 50
- AIJ recovery sequence number is -1
- Area ID number is 2
- Consistency transaction sequence number is 0
- State of page is: corrupt

$
```

The CPT displays information sorted by area and page and includes the following information:

- **AIJ recovery sequence number**
The AIJ recovery sequence number is the AIJ sequence number of the .aij file needed to start recovery for the page. This field is only meaningful for inconsistent pages; hence, a value of -1 is not meaningful. After a page is restored, it is considered inconsistent if the .aij files are not applied. Use this value to apply the first .aij file needed to recover the page.
- **Area ID number**
The storage area ID of the area to which the page belongs.
- **Consistency transaction sequence number**
The transaction sequence number (TSN) of the last committed transaction to the page. This field is only meaningful for inconsistent pages; hence, a value of 0 is not meaningful. After a page is restored, and is still inconsistent, this value will indicate the value of the last committed TSN for the page.
- **State of the page**

The page can be marked as either corrupt or inconsistent.

As areas or pages are restored and recovered, eliminating the source of the corruption, values for these entries are updated in the CPT. Once the page is considered consistent, the entry is removed from the CPT. See Section 8.7 for examples of restoring and recovering database pages that show how values change for entries in the CPT as pages are restored and recovered.

Example 5–5 and Example 5–6 show the attempts to detect checksum and data integrity corruption. No corruption is found in either instance from a scan of the log file. The examples in Section 5.11 show verification results where page-level corruption is detected, and display the type of error messages you receive from this problem.

5.10.1 Using a Checksum Verification to Detect Page Corruption

To see if any information has changed on a page as a result of corruption due to a disk or system problem, use only a page checksum verify operation. Use the RMU Verify command with the following qualifiers:

- Areas
- Checksum_Only
- Log
- Output

For example, when you use the RMU Verify command as shown in Example 5–5, all database page checksums are examined for possible page corruption.

Example 5-5 Verifying Only Database Page Checksums

```
$ RMU/VERIFY/AREAS=*/CHECKSUM_ONLY/LOG DB_DISK:[MFPERS]MF_PERSONNEL
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-DBBOUND, bound to database "DB_DISK:[MFPERS]MF_PERSONNEL.RDB;1"
%RMU-I-OPENAREA, opened storage area RDB$SYSTEM for protected retrieval
%RMU-I-BGNAIPVER, beginning AIP pages verification
%RMU-I-ENDAIPVER, completed AIP pages verification
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-I-ENDABMSPM, completed ABM pages verification
%RMU-I-BGNSEGPAG, beginning verification of RDB$SYSTEM storage area
%RMU-I-ENDSEGPAG, completed verification of RDB$SYSTEM storage area
elapsed time : 00:00:04.66
%RMU-I-OPENAREA, opened storage area EMPIDS_LOW for protected retrieval
%RMU-I-BGNSEGPAG, beginning verification of EMPIDS_LOW storage area
%RMU-I-ENDSEGPAG, completed verification of EMPIDS_LOW storage area
elapsed time : 00:00:00.35
%RMU-I-OPENAREA, opened storage area EMPIDS_MID for protected retrieval
%RMU-I-BGNSEGPAG, beginning verification of EMPIDS_MID storage area
%RMU-I-ENDSEGPAG, completed verification of EMPIDS_MID storage area
elapsed time : 00:00:00.36
%RMU-I-OPENAREA, opened storage area EMPIDS_OVER for protected retrieval
%RMU-I-BGNSEGPAG, beginning verification of EMPIDS_OVER storage area
%RMU-I-ENDSEGPAG, completed verification of EMPIDS_OVER storage area
elapsed time : 00:00:00.35
%RMU-I-OPENAREA, opened storage area DEPARTMENTS for protected retrieval
%RMU-I-BGNSEGPAG, beginning verification of DEPARTMENTS storage area
%RMU-I-ENDSEGPAG, completed verification of DEPARTMENTS storage area
elapsed time : 00:00:00.19
%RMU-I-OPENAREA, opened storage area SALARY_HISTORY for protected retrieval
%RMU-I-BGNSEGPAG, beginning verification of SALARY_HISTORY storage area
%RMU-I-ENDSEGPAG, completed verification of SALARY_HISTORY storage area
elapsed time : 00:00:00.87
%RMU-I-OPENAREA, opened storage area JOBS for protected retrieval
%RMU-I-BGNSEGPAG, beginning verification of JOBS storage area
%RMU-I-ENDSEGPAG, completed verification of JOBS storage area
elapsed time : 00:00:00.19
```

(continued on next page)

Example 5–5 (Cont.) Verifying Only Database Page Checksums

```
%RMU-I-OPENAREA, opened storage area EMP_INFO for protected retrieval
%RMU-I-BGNSEGPAG, beginning verification of EMP_INFO storage area
%RMU-I-ENDSEGPAG, completed verification of EMP_INFO storage area
elapsed time : 00:00:00.21
%RMU-I-OPENAREA, opened storage area RESUME_LISTS for protected retrieval
%RMU-I-BGNSEGPAG, beginning verification of RESUME_LISTS storage area
%RMU-I-ENDSEGPAG, completed verification of RESUME_LISTS storage area
elapsed time : 00:00:00.65
%RMU-I-OPENAREA, opened storage area RESUMES for protected retrieval
%RMU-I-BGNSEGPAG, beginning verification of RESUMES storage area
%RMU-I-ENDSEGPAG, completed verification of RESUMES storage area
elapsed time : 00:00:00.25
%RMU-I-CLOSAREAS, releasing protected retrieval lock on all storage areas
%RMU-S-ENDVERIFY, elapsed time for verification : 0 00:00:21.93
$
```

The checksum verify operation proceeds much faster than a full verify operation because it checks only the page checksum for each database page. It can detect any page corruption at the bit-value level because if the page checksum value does not correspond to the sum of the values for the page, then the page is considered corrupt.

Using the checksum verify operation, you reduce the degree of verification done on a database page but allow the operation to check all database pages. You can also restrict the verification to one or more specific storage areas where you suspect a problem. In this instance, you might make further trade-offs between performance and thoroughness by doing a more thorough check because you are concentrating efforts to one or a few storage areas.

You can also specify the Incremental, Start, and End qualifiers with the Checksum_Only qualifier.

If the RMU Verify Checksum_Only command detects a problem with a certain page, the corrupt page table (CPT) is updated. The best corrective action for these kinds of problems is to restore and recover the pages marked as corrupt.

5.10.2 Detecting a Data Integrity Corruption

If you suspect a problem with the data integrity of your database, you can perform a constraint verification. All constraints defined for your database are checked as shown in Example 5–6. The constraint verification is a complete check of all (possible) defined column and table constraints, including:

- PRIMARY KEY constraints
- NOT NULL constraints

- UNIQUE constraints
- CHECK (predicate) constraint
- Foreign key constraints

Consequently, any defined table or column constraint clauses are checked to see that the data integrity for those tables and columns is valid.

To determine what is being checked during a constraint verify operation, you can use the RDMS\$DEBUG_FLAGS “Sn” flag or RDB_DEBUG_FLAGS to show the optimization strategy with the constraint names and to specify an output file, and use the RDMS\$DEBUG_FLAGS_OUTPUT logical name or RDB_DEBUG_FLAGS_OUTPUT configuration parameter to direct this optimization strategy information to an output file. Specify the Output= qualifier to direct the results of the verify operation to another output file. Thus, output from two operations is directed to two different files. Example 5–6 shows the output as you might see it displayed to the screen. To disable the RDMS\$DEBUG_FLAGS and RDMS\$DEBUG_FLAGS_OUTPUT logical names, deassign them. To disable the RDB_DEBUG_FLAGS and RDB_DEBUG_FLAGS_OUTPUT configuration parameters, remove them from the configuration file.

Example 5–6 Verifying the Constraints for a Database

```
$ DEFINE RDMS$DEBUG_FLAGS "Sn"
$ DEFINE RDMS$DEBUG_FLAGS_OUTPUT CONSTRAINT_CHECK.LIS
$ RMU/VERIFY/CONSTRAINTS /LOG DB_DISK:[MFPERS]MF_PERSONNEL
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-BGNVCONST, beginning verification of constraints for database
.
.
.
~S: Constraint name  WORK_STATUS_PRIMARY_STATUS_CODE
Cross block of 2 entries
Cross block entry 1
  Get      Retrieval sequentially of relation WORK_STATUS
Cross block entry 2
  Conjunct      Aggregate-F2      Conjunct      Get
  Retrieval sequentially of relation WORK_STATUS
~S: Constraint name  STATUS_NAME_VALUES      Conjunct      Get
Retrieval sequentially of relation WORK_STATUS
~S: Constraint name  STATUS_TYPE_VALUES      Conjunct      Get
Retrieval sequentially of relation WORK_STATUS
```

(continued on next page)

Example 5-6 (Cont.) Verifying the Constraints for a Database

```
~S: Constraint name EMPLOYEES_PRIMARY_EMPLOYEE_ID      Conjunct
Match
Outer loop
  Index only retrieval of relation EMPLOYEES
  Index name EMP_EMPLOYEE_ID [0:0]
  Inner loop (zig-zag)
    Aggregate-F2 Index only retrieval of relation EMPLOYEES
    Index name EMP_EMPLOYEE_ID [0:0]
~S: Constraint name EMP_SEX_VALUES      Conjunct      Get
Retrieval sequentially of relation EMPLOYEES
~S: Constraint name EMP_STATUS_CODE_VALUES      Conjunct      Get
Retrieval sequentially of relation EMPLOYEES
~S: Constraint name JOBS_PRIMARY_JOB_CODE
Cross block of 2 entries
  Cross block entry 1
  Get Retrieval sequentially of relation JOBS
  Cross block entry 2
  Conjunct Aggregate-F2 Conjunct      Get
  Retrieval sequentially of relation JOBS
~S: Constraint name WAGE_CLASS_VALUES      Conjunct      Get
Retrieval sequentially of relation JOBS
~S: Constraint name DEPARTMENTS_PRIMARY1      Conjunct
Match
Outer loop
  Index only retrieval of relation DEPARTMENTS
  Index name DEPARTMENTS_INDEX [0:0]
  Inner loop (zig-zag)
    Aggregate-F2 Index only retrieval of relation DEPARTMENTS
    Index name DEPARTMENTS_INDEX [0:0]
~S: Constraint name JOB_HISTORY_FOREIGN1      Conjunct
Match
Outer loop
  Index only retrieval of relation JOB_HISTORY
  Index name JH_EMPLOYEE_ID [0:1]
  Inner loop (zig-zag)
    Aggregate-F1 Conjunct Index only retrieval of relation EMPLOYEES
    Index name EMP_EMPLOYEE_ID [0:0]
~S: Constraint name JOB_HISTORY_FOREIGN2      Conjunct
Match
Outer loop
  Sort Conjunct      Get
  Retrieval sequentially of relation JOB_HISTORY
  Inner loop
  Aggregate Sort Conjunct      Get
  Retrieval sequentially of relation JOBS
```

(continued on next page)

Example 5-6 (Cont.) Verifying the Constraints for a Database

```
~S: Constraint name JOB_HISTORY_FOREIGN3
Cross block of 2 entries
  Cross block entry 1
    Conjunct      Get      Retrieval sequentially of relation JOB_HISTORY
  Cross block entry 2
    Conjunct      Aggregate-F1      Conjunct
    Index only retrieval of relation DEPARTMENTS
      Index name DEPARTMENTS_INDEX [1:1]      Direct lookup
~S: Constraint name SALARY_HISTORY_FOREIGN1      Conjunct
Match
  Outer loop
    Index only retrieval of relation SALARY_HISTORY
      Index name SH_EMPLOYEE_ID [0:1]
  Inner loop      (zig-zag)
    Aggregate-F1      Conjunct      Index only retrieval of relation EMPLOYEES
      Index name EMP_EMPLOYEE_ID [0:0]
~S: Constraint name COLLEGES_PRIMARY_COLLEGE_CODE      Conjunct
Match
  Outer loop
    Index only retrieval of relation COLLEGES
      Index name COLL_COLLEGE_CODE [0:0]
  Inner loop      (zig-zag)
    Aggregate-F2      Index only retrieval of relation COLLEGES
      Index name COLL_COLLEGE_CODE [0:0]
~S: Constraint name DEGREES_FOREIGN1      Conjunct
Match
  Outer loop
    Index only retrieval of relation DEGREES
      Index name DEG_EMP_ID [0:1]
  Inner loop      (zig-zag)
    Aggregate-F1      Conjunct      Index only retrieval of relation EMPLOYEES
      Index name EMP_EMPLOYEE_ID [0:0]
~S: Constraint name DEGREES_FOREIGN2      Conjunct
Match
  Outer loop
    Index only retrieval of relation DEGREES
      Index name DEG_COLLEGE_CODE [0:1]
  Inner loop      (zig-zag)
    Aggregate-F1      Conjunct      Index only retrieval of relation COLLEGES
      Index name COLL_COLLEGE_CODE [0:0]
~S: Constraint name DEG_DEGREE_VALUES      Conjunct      Get
Retrieval sequentially of relation DEGREES
~S: Constraint name CANDIDATES_LAST_NAME_NOT_NULL      Conjunct      Get
Retrieval sequentially of relation CANDIDATES
```

(continued on next page)

Example 5–6 (Cont.) Verifying the Constraints for a Database

```
~S: Constraint name RESUMES_UNIQUE_EMPLOYEE_ID
Cross block of 2 entries
  Cross block entry 1
    Get      Retrieval sequentially of relation RESUMES
  Cross block entry 2
    Conjunct      Aggregate-F2      Conjunct      Get
    Retrieval sequentially of relation RESUMES
~S: Constraint name RESUMES_FOREIGN1
Cross block of 2 entries
  Cross block entry 1
    Conjunct      Get      Retrieval sequentially of relation RESUMES
  Cross block entry 2
    Conjunct      Aggregate-F1      Get
    Retrieval by index of relation EMPLOYEES
      Index name EMPLOYEES_HASH [1:1] Bool Direct lookup
%RMU-I-ENDVCONST, completed verification of constraints for database
%RMU-I-DBBOUND, bound to database "DB_DISK:[MFPERS]MF_PERSONNEL.RDB;1"
%RMU-I-OPENAREA, opened storage area RDB$SYSTEM for protected retrieval
%RMU-I-BGNAIPVER, beginning AIP pages verification
%RMU-I-ENDAIPVER, completed AIP pages verification
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-I-ENDABMSPM, completed ABM pages verification
%RMU-I-CLOSAREAS, releasing protected retrieval lock on all storage areas
%RMU-S-ENDVERIFY, elapsed time for verification : 0 00:00:25.76
$ DEASSIGN RDMS$DEBUG_FLAGS_OUTPUT
$ DEASSIGN RDMS$DEBUG_FLAGS
```

Constraint verification uses an optimization scheme that permits more than one type of constraint to be checked in consecutive scans of the data from the same table when this is the most efficient method. For example, for the DEGREES table, there are three table- or column-level constraints defined: two foreign keys are defined, one of which is the EMPLOYEE_ID column that refers to the primary key, EMPLOYEE_ID of the EMPLOYEES table, and the other is the COLLEGE_CODE column that refers to the primary key, COLLEGE_CODE of the COLLEGE table; the third constraint is a column constraint for the DEGREE column that it contain only one of the following valid values: BA, BS, MA, MS, AA, PhD, or NULL. The constraint verify operation is optimized to check each of these constraints in succession before checking constraints for the next table. In Example 5–6, the following constraint checks are performed for the mf_personnel database in this approximate order:

- For the WORK_STATUS table, the verification operation checks that values for all rows in the STATUS_CODE column correspond to its PRIMARY KEY table-level constraint definition. For all rows in the WORK_STATUS

table, the verification operation checks the value for the STATUS_NAME column is ACTIVE, INACTIVE, or NULL. For all rows in the WORK_STATUS table, the verification operation checks the value for the STATUS_TYPE column is RECORD EXPIRED, FULL TIME, PART TIME, or NULL.

- For the EMPLOYEES table, the verification operation checks that values for all rows in the EMPLOYEES_ID column correspond to its PRIMARY KEY table-level constraint definition. For all rows in the EMPLOYEES table, the verification operation checks the value for the SEX column is M, F, or NULL. For all rows in the EMPLOYEES table, the verification operation checks the value for the STATUS_CODE column is 0, 1, 2, or N.
- For the JOBS table, the verification operation checks that values for all rows in the JOB_CODE column correspond to its PRIMARY KEY table-level constraint definition. For all rows in the JOBS table, the verification operation checks the value for the WAGE_CLASS is 1, 2, 3, 4, or NULL.
- For the DEPARTMENTS table, the verification operation checks that values for all rows in the DEPARTMENT_CODE column correspond to the PRIMARY KEY table-level constraint definition.
- For the JOB_HISTORY table, the verification operation checks that when it refers to the EMPLOYEES table, the foreign key, the EMPLOYEES_ID column in the JOB_HISTORY table, is defined as a UNIQUE or PRIMARY KEY constraint in the EMPLOYEES table.
- For the JOB_HISTORY table, the verification operation checks that when it refers to the JOBS table, the foreign key, the JOB_CODE column in the JOB_HISTORY table, is defined as a UNIQUE or PRIMARY KEY constraint in the JOBS table.
- For the JOB_HISTORY table, the verification operation checks that when it refers to the DEPARTMENTS table, the foreign key, the DEPARTMENT_CODE column in the JOB_HISTORY table, is defined as a UNIQUE or PRIMARY KEY constraint in the DEPARTMENTS table.
- For the SALARY_HISTORY table, the verification operation checks that when it refers to the EMPLOYEES table, the foreign key, the EMPLOYEES_ID column in the SALARY_HISTORY table, is defined as a UNIQUE or PRIMARY KEY constraint in the EMPLOYEES table.
- For the COLLEGES table, the verification operation checks that values for all rows in the COLLEGE_CODE column correspond to its PRIMARY KEY table-level constraint definition.

- For the DEGREES table, the verification operation checks that when it refers to the EMPLOYEES table, the foreign key, the EMPLOYEES_ID column in the DEGREES table, is defined as a UNIQUE or PRIMARY KEY constraint in the EMPLOYEES table.
- For the DEGREES table, the verification operation checks that when it refers to the COLLEGES table, the foreign key, the COLLEGE_CODE column in the DEGREES table, is defined as a UNIQUE or PRIMARY KEY constraint in the COLLEGES table.
- For the DEGREES table, the verification operation checks that the DEGREE column contains one of the following valid values: BA, BS, MA, MS, AA, PhD, or NULL.
- For the CANDIDATES table, the verification operation checks that for all rows in the table, the LAST_NAME column is not null.
- For the RESUMES table, the verification operation checks that when it refers to the EMPLOYEES table, the foreign key, the EMPLOYEE_ID column in the RESUMES table, is defined as a UNIQUE or PRIMARY key constraint in the EMPLOYEES table.

The constraint verify operation checks all rows in only those tables for which you have defined column and table constraints.

5.10.3 Summary of RMU Verify Command Qualifiers for Troubleshooting

In summary, use the following RMU Verify command qualifiers in the following ways to help you troubleshoot problems:

- For detecting suspected database problems:
 - RMU Verify All
Use this command regularly to perform a full verification of your database that includes a data record, a list data fetch check, and a list data verification. If this verify operation detects a problem, some additional troubleshooting may be necessary to isolate the problem, using methods described in Section 5.11.
 - RMU Verify Root Constraint Indexes Nodata Areas=* Snapshots Lareas=*
Use this command for troubleshooting suspected problems with your database because this command does not include a data record fetch check, and its performance is better than a full verification (RMU Verify All) of the database.

- For detecting list data problems:
 - **RMU Verify Lareas=* Segmented_Strings**
Use this command for troubleshooting suspected problems with list data. This command does full verification of segmented strings for all columns for all tables in the database.
- For detecting disk problems and system problems:
 - **RMU Verify areas=* Checksum_Only**
Use this command for troubleshooting disk and system problems because it performs only a checksum verification of pages of all or a specified number of storage areas. If the problem is an unknown change to a page, the checksum check is the best performing command to use to detect these types of problems. If you find a problem with a certain page, the corrupt page table (CPT) is updated. The best corrective measure is to restore and recover those pages that are marked as corrupt. This command also updates and removes a corrupt page entry from the CPT as each corrupt page is restored and each inconsistent page is recovered. See Section 5.10 for more information.
- For detecting index problems:
 - **RMU Verify Index=(index-name) Data**
Use this command for troubleshooting a suspected problem with a specific index and to check that data records can be fetched.
 - **RMU Verify Index=(index-name) Nodata**
Use this command for troubleshooting a suspected problem with a specific sorted index. If you are verifying a large sorted index and you do not want to perform a data record fetch check for improved performance, use the Nodata qualifier. Otherwise, you should always use the Data qualifier.
- For detecting data integrity problems:
 - **RMU Verify Constraints**
Use this command for troubleshooting a suspected data integrity problem; this command checks all defined table and column constraints, including references to external functions and entry points.
- For detecting external function problems:
 - **RMU Verify Functions**

Use this command for troubleshooting a suspected problem with external functions. This command checks all external functions and verifies that the shareable image exists on the system, is located where expected, is accessible, and that all entry points are correct.

5.11 Examples of Database Corruption

Sections 5.11.1, 5.11.2, and 5.11.3 show the results of performing specific verify operations on the database. The sections describe the following types of problems respectively:

- A line index corruption
- A logical area corruption
- A data integrity corruption

For each example, the command used is shown along with a portion of the log file that shows the corruption and the error messages that describe the type of corruption. In each of these sections, the verify strategy used to detect the problem is described along with interpretations of the error and warning messages produced.

5.11.1 Line Index Corruption

In an attempt to isolate a problem, Example 5–7 shows a page checksum bad warning message and a GAPONPAGE error message on page 2 (DEPARTMENTS storage area) that is returned from an area verification of all storage areas in the database. This corrupt page is also logged in the CPT.

Example 5–7 Page Checksum Bad Warning Message and GAPONPAGE Error Message Returned from a DEPARTMENTS Area Verify Operation

```
$ RMU/VERIFY/AREAS=* MF_PERSONNEL
%RMU-W-PAGCKSBAD, area DEPARTMENTS, page 2
    contains an invalid checksum
    expected: 5464C517, found: 546CC517
%RMU-E-GAPONPAGE, unaccounted gap on page 2
    free space end offset : 0000004E (hex)
    minimum offset of any line : 00000206 (hex)
```

Use the online Help facility for Oracle RMU, specifically RMU_ERRORS, to find a more detailed explanation of this error message. For example, for the error message GAPONPAGE, the information displays as shown in Example 5–8.

Example 5–8 Online Help File Explanation for the RMU Verify GAPONPAGE Error Message

```
RMU_ERRORS
```

```
GAPONPAGE
```

```
unaccounted gap on page <num>  
  free space end offset : <num> (hex)  
  minimum offset of any line : <num> (hex)
```

```
Explanation: A gap was found between the end of free space and  
the beginning of the line closest to the beginning of the page.  
This could be caused by the corruption of locked free space  
length, free space length or the line index.
```

```
User Action: Dump the page in question to determine the  
corruption. Restore the database and verify again.
```

The GAPONPAGE error message can be interpreted as follows: the minimum offset value for any line is 206 hex. The first record on the page, in this case, line 2, is located at offset 206 hex. A display of page 2 in Example 5–9 shows an offset of 206 hex for line 2, which is the minimum offset value of the three lines listed on that page. The free space offset end value is at offset 4E hex. In Example 5–9, the line 2: TSN index ends at offset 002F. To this value add the length of free space 1E hex to get the calculated value 4E hex or, the free space offset end value. Between offset 4E hex and offset 206 hex, the verify operation has detected an unaccounted-for gap. Example 5–9 displays the corrupt portion of page 2. All data beginning at offset 8E hex and continuing to offset 205 hex is labeled as **** junk ****. This is part of the unaccounted-for gap.

Example 5-9 Corrupted Page 2 of the DEPARTMENTS Storage Area

```

$ RMU/DUMP/AREAS=DEPARTMENTS /START=2 /END=2 MF_PERSONNEL
.
.
.
0005 00000002 0000 page 2, physical area 5
546CC517 0006 checksum = 546CC517
0091FAC0 9EA7B160 000A time stamp = 31-MAR-1995 09:08:02.55
0000 001E 0012 30 free bytes, 0 locked
0003 0016 3 lines
0005 03E4 0018 line 0: offset 03E4, 5 bytes
01AE 0236 001C line 1: offset 0236, 430 bytes
0030 0206 0020 line 2: offset 0206, 48 bytes

002F01D6 0024 line 0: TSN 3080662
003101A4 0028 line 1: TSN 3211684
002E0176 002C line 2: TSN 3015030
696261434D42434D0001280000010018 008E ** junk ** '.....(.MCBMCabi'
756E614D20656D61724620262074656E 009E ** junk ** 'net & Frame Manu'
87353931303020676E69727574636166 00AE ** junk ** 'facturing 00195.'
534D424D00011E000001001800F80000 00BE ** junk ** '..ø.....MBMS'
727574636166756E614D206472616F42 00CE ** junk ** 'Board Manufactur'
363130300420846874756F5320676E69 00BE ** junk ** 'ing South. .0016'
424D00011E000001001800F800008736 00EE ** junk ** '6...ø.....MB'
74636166756E614D206472616F424E4D 00FE ** junk ** 'MNBoard Manufact'
30300420846874726F4E20676E697275 010E ** junk ** 'uring North. .00'
000118000001001800F8000087343631 011E ** junk ** '164...ø.....'
6166756E614D206472616F42464D424D 012E ** junk ** 'MBMFBoard Manufa'
87373232303004208A676E6972757463 013E ** junk ** 'cturing. .00227.'
20474E45000110000001001800F80000 014E ** junk ** '..ø.....ENG '
3030042092676E697265656E69676E45 015E ** junk ** 'Engineering. .00'
00011B000001001800F8000087313734 016E ** junk ** '471...ø.....'
45206C6163696E616863654D434D4C45 017E ** junk ** 'ELMCMechanical E'
313030042087676E697265656E69676E 018E ** junk ** 'ngineering. .001'
4C4500011E0000010018F80000873039 019E ** junk ** '90...ø.....EL'
20736D657473795320656772614C5347 01AE ** junk ** 'GSLarge Systems '
3030042084676E697265656E69676E45 01BE ** junk ** 'Engineering. .00'
00011C000001001800F8000087393633 01CE ** junk ** '369...ø.....'
207363696E6F727463656C454C454C45 01DE ** junk ** 'ELELElectronics '
3030042086676E697265656E69676E45 01EE ** junk ** 'Engineering. .00'
00F8000087383831 01FE ** junk ** '188...ø.'

0018 0206 line 2: record type 24
00 0001 0208 Control information
.... 43 bytes of static data
657461726F70726F434E4D444100011D 020B data '...ADMNCorporate'
856E6F697461727473696E696D644120 021B data ' Administration.'
F800008735323230300420 022B data ' .00225...ø'

```

(continued on next page)

Example 5–9 (Cont.) Corrupted Page 2 of the DEPARTMENTS Storage Area

```
      .... total B-tree node size: 430
003F 2003 0236 line 1: index node for set 64
002A FFFFFFFF FFFF 023A owner 65:-1:-1
      00D6 0242 214 bytes of entries
      8200 0244 level 1, full suffix
      00 05 0246 5 bytes stored, 0 byte prefix
4E4D444100 0248 key '.ADMN'
      1633 06 024D pointer 65:2:2
      01 04 0250 4 bytes stored, 1 byte prefix
      00 pfx '.'
4C454C45 0252 key 'ELEL'
      1634 06 0256 pointer 65:2:3
.
.
.
```

Because there appears to be valid data in this unaccounted-for gap between offset 4E hex and offset 206 hex, you can conclude that there is something wrong with the line index portion of the page. Example 5–10 displays the same but uncorrupted portion of page 2. The number of lines listed is 11 in Example 5–10, but on the corrupted page, it is 3, which shows the corruption. To correct this problem, restore and recover page 2 of the DEPARTMENTS area and verify it again.

Example 5–10 Uncorrupted Page 2 of the DEPARTMENTS Storage Area

```
$ RMU/DUMP/AREAS=DEPARTMENTS /START=2 /END=2 MF_PERSONNEL
.
.
.
      0005 00000002 0000 page 2, physical area 5
      546CC517 0006 checksum = 546CC517
0091FAC0 9EA7B160 000A time stamp = 31-MAR-1995 09:08:02.55
      0000 001E 0012 30 free bytes, 0 locked
      000B 0016 11 lines
      0005 03E4 0018 line 0: offset 03E4, 5 bytes
      01AE 0206 001C line 1: offset 0206, 430 bytes
      0030 03B4 0020 line 2: offset 03B4, 48 bytes
      002F 01D6 0024 line 3: offset 01D6, 47 bytes
      0031 01A4 0028 line 4: offset 01A4, 49 bytes
      002E 0176 002C line 5: offset 0176, 46 bytes
      0023 0152 0030 line 6: offset 0152, 35 bytes
      002B 0126 0034 line 7: offset 0126, 43 bytes
      0031 00F4 0038 line 8: offset 00F4, 49 bytes
      0031 00C2 003C line 9: offset 00C2, 49 bytes
      002A 008E 0040 line 10: offset 008E, 51 bytes

      00000000 0044 line 0: TSN 0
      00000028 0048 line 1: TSN 40
      00000028 004C line 2: TSN 40
      00000028 0050 line 3: TSN 40
      00000028 0054 line 4: TSN 40
      00000028 0058 line 5: TSN 40
      00000028 005C line 6: TSN 40
      00000028 0060 line 7: TSN 40
      00000028 0064 line 8: TSN 40
      00000028 0068 line 9: TSN 40
      00000028 006C line 10: TSN 40

00000000000000000000000000000000000000000000000000000000000000000000 0070 free space '.....'
      00000000000000000000000000000000000000000000000000000000000000000000 0080 free space '.....'

      0018 008E line 10: record type 24
      00 0001 0090 Control information
      .... 46 bytes of static data
262074656E696261434D42434D000128 0093 data '(..MCBMCabinet &'
7574636166756E614D20656D61724620 00A3 data ' Frame Manufactu'
      F8000087353931303020676E6972 00B3 data 'ring 00195...ø'
      00 00C1 padding '.'
```

(continued on next page)

Example 5–10 (Cont.) Uncorrupted Page 2 of the DEPARTMENTS Storage Area

```

0018 00C2 line 9: record type 24
00 0001 00C4 Control information
      .... 44 bytes of static data
6E614D206472616F42534D424D00011E 00C7 data '...MBMSBoard Man'
6874756F5320676E69727574636166675 00D7 data 'ufacturing South'
      F80000873636313030042084 00E7 data '. .00166...ø'
00 00F3 padding '.'
.
.
.
      .... total B-tree node size: 430
003F 2003 0236 line 1: index node for set 64
002A FFFFFFFF FFFF 023A owner 65:-1:-1
      00D6 0242 214 bytes of entries
      8200 0244 level 1, full suffix
      00 05 0246 5 bytes stored, 0 byte prefix
4E4D444100 0248 key '.ADMN'
      1633 60 024D pointer 65:2:2
      01 04 0250 4 bytes stored, 1 byte prefix
      00 pfx '.'
4C454C45 0252 key 'ELEL'
      1634 60 0256 pointer 65:2:3
.
.
.

```

5.11.2 Logical Area Corruption

When a logical area (index) becomes corrupt due to a software error or system failure, you can receive an access violation or bugcheck dump. You should perform an RMU Verify operation and specify the Lareas qualifier to pinpoint the problem.

Example 5–11 shows this type of problem. The problem has been traced to page 2 in the DEPARTMENTS storage area and the verify operation will log this page in the CPT.

When you perform an RMU Verify Larea command on all logical areas, a page checksum bad warning message is returned. This message tells you that information on page 2 is corrupt.

Example 5–11 Page Checksum Bad Warning Message Returned from a Checksum Verify Operation of the DEPARTMENTS Logical Area

```
$ RMU/VERIFY/LAREA=* MF_PERSONNEL
%RMU-W-PAGCKSBAD, area DEPARTMENTS, page 2
    contains an invalid checksum
    expected: 546CCD17, found: 546CC517
```

To determine what other logical areas are located in the DEPARTMENTS storage area, you can use the RMU Dump command and inspect the database pages, or you can use the RMU Analyze Areas=Departments command to list the logical areas. Two logical areas are in the DEPARTMENTS storage area: the DEPARTMENTS rows and the sorted index structures for the DEPARTMENTS_INDEX index. To pinpoint the problem, use the RMU Verify Index command to verify this sorted index and specify the Nodata qualifier. Example 5–12 shows that on line 3, a storage record in the B-tree node with the prefix key “ADMN” contained a bad prefix key length, expected 5, found 76. The warning message indicates that the B-tree node keys are also not in lexical or alphabetical order. These informational and warning messages show that the DEPARTMENTS_INDEX sorted index is corrupt.

Example 5–12 B-Tree Lexical Error Returned from an Index Verify Operation with No Data Record Check

```
$ RMU/VERIFY/INDEX=DEPARTMENTS_INDEX /NODATA MF_PERSONNEL
%RMU-I-BTRPFERR, area RDB$SYSTEM, page 828651012, line 3
    storage record B-TREE NODE, with prefix key "ADMN"
    contains a bad prefix key len, expected: 5, found: 76
%RMU-W-BTRLEXERR, b-tree node keys not in lexical order
    found key ".ADMN.....!.....@...fð,.....| Ø.â...â.k#..
l.â.ãð,.....tEL'4...GS'5...MC'6...NG `7...MBMfA.c...Na.c...Sa.c...CB
Ma.c...S`B...G" in b-tree node at 64:2:1
    followed by key ".ADMN.....!.....@...fð,.....| Ø.â...
â.k#..l.â.ãð,.....TG" in b-tree node at 64:2:1
%RMU-I-BTRROODBK, root dbkey of B-tree is 64:2:1
$
```

When you repeat the index verification and accept the default Data qualifier to check the data record fetch, additional messages display, as shown in Example 5–13. The first error message indicates that line index entry 5682 cannot be found; either the line index does not exist or the database key (dbkey) pointer is wrong. Other E level messages indicate that there was an error fetching dbkey 43:3:5682 and that the data record cannot be fetched from the B-tree index node. The I level messages indicate that the problem is with a B-tree node record and that there is a bad prefix key length: expected 5, found 76. The warning message indicates that the B-tree node keys are not in lexical order.

Example 5–13 B-Tree Lexical Error Returned from a Checksum Verify Operation with a Data Record Check

```
$ RMU/VERIFY/INDEX=DEPARTMENTS_INDEX MF_PERSONNEL
%RMU-E-LNGTRLNDX, line 5682 beyond line index on page
%RMU-E-BADDBKFET, error fetching dbkey 43:3:5682
%RMU-E-ERRDATFET, error fetching data record from B-tree index node
%RMU-I-BTRNODDBK, dbkey of B-tree node for data record is 64:2:1
%RMU-I-BTRPFERR, area RDB$SYSTEM, page 828651012, line 3
    storage record B-TREE NODE, with prefix key "ADMN"
    contains a bad prefix key len, expected: 5, found: 76
```

(continued on next page)

Example 5–13 (Cont.) B-Tree Lexical Error Returned from a Checksum Verify Operation with a Data Record Check

```
%RMU-E-BADDBKFET, error fetching dbkey 65:828651012:3
%RMU-E-ERRDATFET, error fetching data record from B-tree index node
%RMU-I-BTRNODDBK, dbkey of B-tree node for data record is 64:2:1
%RMU-W-BTRLEXERR, b-tree node keys not in lexical order
                    found key ".ADMN.....9.....@...fð,.....| Ø.â...â.k#..
l.â.ãð,.....tEL'4...GS'5...MC'6...NG `7...MBMfa.c...Na.c...Sa.c...CB
Ma.c...S'B...G" in b-tree node at 64:2:1
                    followed by key ".ADMN.....9.....@...fð,.....| Ø.â...
â.k#..l.â.ãð,.....TG" in b-tree node at 64:2:1
%RMU-I-BTRROOBBK, root dbkey of B-tree is 64:2:1
$
```

In Example 5–13, DEPARTMENTS_INDEX index is corrupt and the dbkey pointer is incorrect. You can choose one of the following options:

- Restore and recover the entire database
- Restore and recover the DEPARTMENTS storage area or only page 2
- Delete the corrupt index and rebuild it

In this instance, if the dbkey pointer appears to be the problem, it may be easier and best to delete the DEPARTMENTS_INDEX sorted index and rebuild it. However, if you detect other problems such as other corrupt pages and corrupted data records on this or other pages, then restore and recover the DEPARTMENTS storage area or any corrupt pages to correct the problems.

Once you fix the problem, you can inspect and compare the information on page 2 in the DEPARTMENTS storage area for both the corrupted and uncorrupted pages to determine exactly what the problem was. Example 5–14 shows the corrupted portion of page 2, and Example 5–15 shows the same portion of the uncorrupted page. On the corrupted page, the dbkey pointer at offset 024D is 43:3:5682 followed by dbkey pointer 65:828651012:3, which is followed by dbkey pointer 65:3:3. On the same but uncorrupt page, the sequence of dbkeys is 65:2:2, 65:2:3, 65:2:4, and so forth. This is the reason for the error messages: line 5682 beyond line index on page and error fetching dbkey 43:3:5682. The dbkey 43:3:5682 does not exist.

Also, the first B-tree node key on the corrupted page is ADMN, followed by EL, GS, NG, Na, CBMa, G, TG, and NFG. The reason for one of the warning messages is that these B-tree node keys are not in lexical or alphabetical order. On the uncorrupted page, these B-tree node keys are in lexical order, ADMN, ELEL, GS, and so forth.

Example 5-14 Corrupted Portion of Page 2 of the DEPARTMENTS Storage Area

```

$ RMU/DUMP/AREA=DEPARTMENTS /START=2 /END=2 MF_PERSONNEL
.
.
.
      .... total B-tree node size: 430
      003F 2003 0236 line 1: index node for set 64
002A FFFFFFFF FFFF 023A owner 45:-1:-1
      00D6 0242 214 bytes of entries
      8200 0244 level 1, full suffix
      00 05 0246 5 bytes stored, 0 byte prefix
      4E4D444100 0248 key '.ADMN'
      01041633 68 024D pointer 43:3:5682
      4C 45 0252 69 bytes stored, 76 byte prefix
801B26230000000000000000000000004E4D444100 pfx '.ADMN.....#&..'
000000000000A034010E00002FFC0000 pfx '..ü/....4.....'
00077C207FE20AAC7FE20ADC2FFC0000 pfx '..ü/Û.â.¬.â. |..'
0301630A61530401016309614E040101 pfx '@...°...æ.....'
00000001001282E6001280B0000001AE pfx 'Æ...2...Ôw..'
      000777D40000A032001280C6
34604C45 0254 key 'EL`4'
03163660434D03021635605347030216 0258 key '...GS`5...MC`6..'
630861464D424D010416376020474E02 0268 key '.NG `7...MBMfa.c'
0301630A61530401016309614E040101 0278 key '...Na.c...Sa.c..'
020316426053040101630B614D424302 0288 key '.CBMa.c...S`B...'
      47 0298 key `G'
      0316436054 56 0299 pointer 65:828651012:3
      4B 02 029F 2 bytes stored, 75 byte prefix
801B26230000000000000000000000004E4D444100 pfx '.ADMN.....#&..'
000000000000A034010E00002FFC0000 pfx '..ü/....4.....'
00077C207FE20AAC7FE20ADC2FFC0000 pfx '..ü/Û.â.¬.â. |..'
0301630A61530401016309614E040101 pfx '@...°...æ.....'
00000001001282E6001280B0000001AE pfx 'Æ...2...Ôw..'
      0777D40000A032001280C6
      4754 02A1 key 'TG'
      1644 60 02A3 pointer 65:3:3
      02 03 02A6 3 bytes stored, 2 byte prefix
      4100 pfx '.A'
      47464E 02A8 key `NFG'
      1645 60 02AB pointer 65:3:4
.
.
.

```

Example 5–15 Uncorrupted Portion of Page 2 of the DEPARTMENTS Storage Area

```
$ RMU/DUMP/AREA=DEPARTMENTS /START=2 /END=2 MF_PERSONNEL
.
.
.
      .... total B-tree node size: 430
      003F 2003 0236 line 1: index node for set 64
002A FFFFFFFF FFFF 023A owner 65:-1:-1
      00D6 0242 214 bytes of entries
      8200 0244 level 1, full suffix
      00 05 0246 5 bytes stored, 0 byte prefix
      4E4D444100 0248 key '.ADMN'
      1633 60 024D pointer 65:2:2
      01 04 0250 4 bytes stored, 1 byte prefix
      00 pfx '.'
      4C454C45 0252 key 'ELEL'
      1634 60 0256 pointer 65:2:3
      03 02 0259 2 bytes stored, 3 byte prefix
      4C4500 pfx '.EL'
      5347 025B key 'GS'
      1635 60 025D pointer 65:2:4
.
.
.
```

5.11.3 Data Integrity Corruption

Example 5–16 shows the log of an area verification. The page warning error messages indicate a problem with the page header. In this case, the page checksum is the problem because it is part of the page header information. All other items checked on the page are fine. The problem may be with an index or with data. Because there are no indexes defined on the table and there are only 15 rows in the table, a check of the data integrity is the next logical step in troubleshooting the problem.

Example 5–16 Page Errors Returned from an Area Verify Operation of the JOBS Storage Area

```
$ RMU/VERIFY/AREAS=* /LOG MF_PERSONNEL
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-DBBOUND, bound to database "DBDISK:[MFPERS]MF_PERSONNEL.RDB;1"
%RMU-I-OPENAREA, opened storage area RDB$SYSTEM for protected retrieval
%RMU-I-BGNAIPVER, beginning AIP pages verification
%RMU-I-ENDAIPVER, completed AIP pages verification
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-I-ENDABMSPM, completed ABM pages verification
%RMU-I-OPENAREA, opened storage area JOBS for protected retrieval
%RMU-I-BGNSEGPAG, beginning verification of JOBS storage area
%RMU-W-PAGCKSBAD, area JOBS, page 2
                    contains an invalid checksum
                    expected: 13BFE1B7, found: 13BFE1C7
%RMU-W-PAGERERRORS,      1 page error encountered
                        1 page header format error
                        0 page tail format errors
                        0 area bitmap format errors
                        0 area inventory format errors
                        0 line index format errors
                        0 segment format errors
                        0 space management page format errors
                        0 differences in space management of data pages
%RMU-I-ENDSEGPAG, completed verification of JOBS storage area
                    elapsed time : 00:00:00.62
%RMU-I-CLOSAREAS, releasing protected retrieval lock on all storage areas
%RMU-S-ENDVERIFY, elapsed time for verification :    0 00:00:09.74
$
```

Example 5–17 shows that the constraint verify operation is failing to access and read area 7, the JOBS storage area, from page 2 through 2 (7:2-2).

Example 5–17 Fatal Error Messages Returned from a Constraint Verify Operation

```
$ RMU/VERIFY/CONSTRAINT MF_PERSONNEL
%RDB-F-IO_ERROR, input or output error
-RDMS-F-CANTREADDBS, error reading pages 7:2-2
-RDMS-F-CHECKSUM, checksum error - computed 13BFE1B7, page contained 13BFE1CF
```

To investigate further, you can define the `RDMS$DEBUG_FLAGS` logical name as “Sn” to display the access strategy with the constraint names and define the `RDMS$DEBUG_FLAGS_OUTPUT` logical name (for OpenVMS systems) or the `RDB_DEBUG_FLAGS_OUTPUT` configuration parameter (for Digital UNIX systems) to place the results in a file, as shown in Example 5–18. Specify the

Log and Output qualifiers to direct the output from the constraint verification to a file. Remember to deassign both logical names when you have completed this constraint verify operation.

Example 5–18 Tracing the Corruption to a Set of Constraints

```

$ DEFINE RDMS$DEBUG_FLAGS "Sn"
$ DEFINE RDMS$DEBUG_FLAGS_OUTPUT CONSTRAINT_CK.LIS
$ RMU/VERIFY/CONSTRAINT /LOG /OUTPUT=MFPERPERS_CONSTRAINT_CK.LIS MF_PERSONNEL
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-BGNVCONST, beginning verification of constraints for database
"DBDISK:[MFPERPERS]MF_PERSONNEL.RDB;1"
.
.
.
~S: Constraint name JOBS_PRIMARY_JOB_CODE
Cross block of 2 entries
Cross block entry 1
Get Retrieval sequentially of relation JOBS
Cross block entry 2
Conjunct Aggregate-F2 Conjunct Get
Retrieval sequentially of relation JOBS
%RDB-F-IO_ERROR, input or output error
-RDMS-F-CANTREADDBS, error reading pages 7:2-2
-RDMS-F-CHECKSUM, checksum error - computed 25211595, page contained 252115A5
%RMU-I-ENDVCONST, completed verification of constraints for database
"DBDISK:[MFPERPERS]MF_PERSONNEL.RDB;1"
%RMU-I-DBBOUND, bound to database "DBDISK:[MFPERPERS]MF_PERSONNEL.RDB;1"
%RMU-I-OPENAREA, opened storage area RDB$SYSTEM for protected retrieval
%RMU-I-BGNAIPVER, beginning AIP pages verification
%RMU-I-ENDAIPVER, completed AIP pages verification
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-I-ENDABMSPM, completed ABM pages verification
%RMU-I-CLOSAREAS, releasing protected retrieval lock on all storage areas
%RMU-S-ENDVERIFY, elapsed time for verification : 0 00:01:09.53
$ DEASSIGN RDMS$DEBUG_FLAGS_OUTPUT
$ DEASSIGN RDMS$DEBUG_FLAGS

```

In Example 5–18, the constraint Verify operation stops at the point where the JOBS table is being accessed and the JOBS_PRIMARY_JOB_CODE constraint is the last constraint to be verified without errors. This indicates a problem with the data being validated for the constraints defined for the JOBS table and specifically the very next constraint. From Example 5–6, you can determine that the constraint that was being verified when the error displayed was WAGE_CLASS_VALUES. Take note of the area and pages being read when the constraint verify operation stopped. That is, the error states error reading pages 7:2-2. When you check the JOBS table constraint definitions,

you find the constraint named WAGE_CLASS_VALUES is a column-level constraint that checks for valid values for the WAGE_CLASS column, which are 1, 2, 3, 4, or NULL. The problem is with one of the column-level constraint values (in the WAGE_CLASS column) for a row stored on this page. At this point, try to identify the exact problem by displaying the page. Example 5–19 displays 2 of the 15 rows in the JOBS table.

Example 5–19 Corrupted Portion of Page 2 of the JOBS Storage Area

```

.
.
.
          0017 0198 line 15: record type 23
00 0001 019A Control information
          .... 34 bytes of static data
65725020656369563444535056000114 019D data '...VPSD4Vice Pre'
E4E1C0007270E0082085746E65646973 01AD data 'sident. .àpr.Àää'
          E000 01BD data '.à'
          00 01BF padding '.'

          0017 01C0 line 14: record type 23
00 0001 01C2 Control information
          .... 37 bytes of static data
20736D6574737953244D475053000123 01C5 data '#..SPGM$Systems '
002625A0202072656D6D6172676F7250 01D5 data 'Programmer .%&.'
          E0004C4B40 01E5 data '@KL.à'

.
.
.

```

Because the JOBS table is small (15 rows), it becomes relatively easy to isolate the problem. The primary key values for JOB_CODE, beginning at offsets 01A0 and 01C8 respectively, are VPSD and SPGM, which are both valid values. However, for the WAGE_CLASS column, the 4 and \$ values display at offsets 01A4 and 01CC respectively. The value \$ is not a valid value. This is the point of the corruption.

The best corrective measure is to use SQL to delete the row and add the row again, or try and update the WAGE_CLASS column value for that row. If you were to restore and recover the area or page, you might not correct the problem if it has persisted for some time. You would have to locate an uncorrupt backup file.

After you have made the correction, verify the JOBS storage area, as shown in Example 5–20. Note that the area verification of the JOBS storage area was successful.

Example 5–20 Verifying the JOBS Storage Area

```
$ RMU/VERIFY/AREAS=JOBS /LOG MF_PERSONNEL
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-DBBOUND, bound to database "DBDISK:[MFPERS]MF_PERSONNEL.RDB;1"
%RMU-I-OPENAREA, opened storage area RDB$SYSTEM for protected retrieval
%RMU-I-BGNAIPVER, beginning AIP pages verification
%RMU-I-ENDAIPVER, completed AIP pages verification
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-I-ENDABMSPM, completed ABM pages verification
%RMU-I-OPENAREA, opened storage area JOBS for protected retrieval
%RMU-I-BGNSEGPAG, beginning verification of JOBS storage area
%RMU-I-ENDSEGPAG, completed verification of JOBS storage area
                    elapsed time : 00:00:00.27
%RMU-I-CLOSAREAS, releasing protected retrieval lock on all storage areas
%RMU-S-ENDVERIFY, elapsed time for verification : 0 00:00:13.11
```

Example 5–21 shows a display of page 2 of the JOBS storage area to determine what the valid value for line 14 at offset 01CC should be. Note the WAGE_CLASS value is 4.

Example 5–21 Uncorrupted Portion of Page 2 of the JOBS Storage Area

```
.
.
.
                0017 01C0 line 14: record type 23
                00 0001 01C2 Control information
                .... 37 bytes of static data
20736D6574737953344D475053000123 01C5 data '#..SPGM4Systems '
002625A0202072656D6D6172676F7250 01D5 data 'Programmer .%&.'
                E0004C4B40 01E5 data '@KL.à'
.
.
.
```

Examples 5–18, 5–19, and 5–20 pinpoint simple constraint verification problems. In reality, it may take considerably more effort to track down the actual constraint or set of constraints where the problem exists. It is best to track the problem to a row or group of rows to try to determine what caused the problem.

There are several important points to keep in mind when you are verifying constraints for your database. With multiple tables, you can define the following logical names and configuration parameters to help you identify problem areas:

- `RDMS$DEBUGS_FLAGS` and `RDMS$DEBUGS_FLAGS_OUTPUT` for OpenVMS systems
- `RDB_DEBUGS_FLAGS` and `RDB_DEBUGS_FLAGS_OUTPUT` for Digital UNIX systems

Because the actual constraint in which the problem exists is never identified, you must identify it yourself. The constraint verify operation stops once an error is detected. More precisely, it stops on the page where the checksum is invalid. Note the name of the last constraint that successfully verified from the failed `RDMS$DEBUGS_FLAGS_OUTPUT` or `RDB_DEBUGS_FLAGS` log, then refer to your successful `RDMS$DEBUGS_FLAGS_OUTPUT` or `RDB_DEBUGS_FLAGS_OUTPUT` log to determine the name of the very next constraint; this is always the one that failed verification.

You must identify the exact problem or problems on the page to prevent them from recurring. For example, is the problem due to a poorly defined trigger definition, an abnormal image termination during a batch-update transaction, a software error, or some other problem? By locating the exact problem, you can trace the cause. If you cannot determine the exact cause and restore the storage area, the problem may recur. The invalid value you find for a specific column in a specific row for a specific table provides you with clues. Is the value that is corrupted due to a random error or is it changing each time?

You can also pinpoint constraint problems by forming queries on the affected table. You can access and read all rows or a specific row and read specific columns for which constraints are defined. A data integrity problem with a specific column for a specific row in the table produces an access violation and bugcheck dump. This information can be useful for pinpointing problems too.

In cases where the database has more than a single failed constraint problem, you will not know this until you verify and solve the first failed constraint verification problem. The constraint verify operation detects only the first constraint that failed, produces an error message, and stops. If your database shows more than one checksum error, note each one.

If you pinpoint the problem as a data integrity problem, you must solve the first problem shown by correcting the problem and performing a constraint verify operation. In recursive fashion, you pinpoint the next data integrity problem, correct it, and perform a constraint verify operation, and so forth.

Finally, when you have corrected all data integrity problems and you have performed one last successful constraint verify operation, then you can be certain that all affected storage areas have data integrity. Thus, if you have multiple data integrity problems in your database, each must be detected, corrected, and verified.

Repairing or Altering a Database

This chapter briefly describes the RMU Repair command to repair specific types of database problems and explains how to use the RMU Alter command, which invokes the RdbALTER utility to patch a database or move it from one device to another.

6.1 Using RMU Repair

You can use the RMU Repair command to correct the following database problems:

- To repair all types of space area management (SPAM) page corruption by reconstructing the SPAM pages in one or more storage areas
- To repair all area bit map (ABM) page format errors
- To repair all page tail errors to the satisfaction of the RMU Verify command
This is done by making sure that every database page is in a logical area and contains the appropriate information for that logical area.
- To correct some performance problems without having to export and import the database
- To initialize free database pages that contain no data for tables in uniform storage areas
- To create and initialize snapshot (.snp) files
- To initialize the transaction sequence numbers (TSNs) to zero in a database
- To update the area inventory page (AIP) for specified logical areas before building SPAM pages
Update information can include new SPAM thresholds for logical areas in uniform storage areas, marking a logical area as deleted, or specifying a different record length to store in a logical area inventory entry.
- To initialize .snp files, relocate them, and change their allocation

- To set null those columns that are found to contain damaged or missing list segments

Damaged or missing list segments may be due to the after-image journal (.aij) file not being enabled for a write-once area on a write-once, read-many (WORM) device and the data becoming inconsistent because of a WORM disk failure that cannot be recovered.

The RMU Repair command cannot correct problems such as checksum errors, corrupted user data, or corrupted indexes. Use the RMU Alter command described in Section 6.2 and in the *Oracle RMU Reference Manual* to correct these problems.

Oracle Rdb recommends that you back up your database or have an exported copy of your database before attempting to repair your database in the event that the repair operation is not effective. Because repair operations to the database are not journaled in the .aij file and so cannot be rolled forward with the RMU Recover command, Oracle Rdb also recommends that you back up your database following a repair operation. Because the database is not flagged as having been repaired in the database root file (.rdb) or database backup file (.rbf), you must assume responsibility for keeping a record of all repair operations.

You must have complete and exclusive access to the database to use the RMU Repair command. For a complete description of the RMU Repair command, privileges required, and examples, see the *Oracle RMU Reference Manual*.

6.2 Using RdbALTER

Oracle Rdb recommends that you use the RMU Alter command *only* under the following conditions:

- To make the database available for further investigation of a problem
- To facilitate permanent error correction activity, such as exporting the database by using the SQL EXPORT statement, using the RMU Extract command to extract database information, or using the RMU Unload command to unload data
- To make the database temporarily available until permanent corrective action can be scheduled

Correct use of the RMU Alter command for permanent repairs requires a very detailed understanding of Oracle Rdb software, of the database design, and of the current state of the database. Moreover, it is rare that a database corruption problem is a single-point data corruption. Problems that generate database corruptions frequently create multiple corruptions in the database

that must be located and corrected. Database corruption that goes undetected for some time frequently introduces secondary related corruptions elsewhere in the database.

With these recommendations in mind, you can use the RdbALTER utility to perform the following tasks:

- Perform low-level patching of corrupted data storage or space area management (SPAM) pages.
- Move database root, area, and snapshot files (in conjunction with the operating system Copy command).
- Move files from one disk to another or from one directory to another on the same disk.
- Make low-level changes to the data or pointers.

You cannot use RdbALTER to alter journal files.

You should use RdbALTER on a working database only if the following conditions exist:

- You fully understand the internal data structures and access mechanisms of a database storage area page (Chapter 12 describes page structure in detail).
- You know the information the database should contain.
- The problems are specific, local, and few.

To patch database pages, perform the following steps:

1. Use the RMU Verify command to locate the source of database corruption.
2. Display the contents of the corrupt storage area by using the RMU Dump command.

You might also need a complete display of the database header (use RMU Dump Header) to obtain the internal ID numbers for storage areas, logical areas, and records. (Refer to Chapter 11 and the *Oracle RMU Reference Manual* for more information on using the RMU Dump command.)

3. Once you locate the source of corruption, determine the correct contents of the corrupted page.

If the problem occurs because of a corrupt database key (dbkey) pointer, determine the correct dbkey pointer. If the problem occurs because of corrupt data, determine the correct data that belongs on that page.

4. After you locate the source of the corruption and determine the correction that must be made, use the RMU Alter command to invoke the RdbALTER utility.
5. Use RdbALTER commands to patch the database.

Usually, restoring a database from recovery-unit journal (.ruj) and .aij files offers a more simple and sure remedy. However, some database structure problems are not apparent until long after they have occurred. Recent backup files might carry the same errors, making the restore and recovery techniques ineffective. (The database backup utility displays a message indicating corruption only if that corruption was caused by a batch-update operation.)

Direct altering of pages is an error-prone procedure that can compound the problem, if misused. Always use the RdbALTER Log command to create a log file of the RdbALTER session when you alter a production database. If you make a mistake, you can use the log file to retrace your steps. If you realize you made a mistake during the current RdbALTER session, simply roll back to undo all the changes. The altered pages are not written to the database until you issue the Commit command.

If you want to add comments to RdbALTER commands, you can use an exclamation point (!) as the comment delimiter. RdbALTER ignores any text you type after the exclamation point. Pressing the Return key marks the end of a command. Events causing informational messages are not considered errors and do not terminate the command line.

The following sections describe the functions of RdbALTER commands and explain when and why to use them.

6.2.1 Attaching to a Database

To access a database, enter the RdbALTER environment, using the RMU Alter command:

```
$ RMU/ALTER  
RdbALTER>
```

At the RdbALTER prompt, type the Attach command to attach to a database. The syntax for the Attach command is as follows:

```
Attach root-file-spec
```

The database that you specify with the *root-file-spec* parameter is now attached to RdbALTER. You can perform an implicit attach to the database by specifying the root file on the RMU Alter command line. You can then alter the pages of its storage area (.rda) files.

When you attach to the database, RdbALTER fetches page 1 of the first storage area. You can select a different storage area and page by using the `Area . . . Page` or `Page` commands. After you enter a `Commit` or `Rollback` command, type the `Detach` command to terminate database access. During the interval of time between the `Attach` and `Detach` commands, you have `Exclusive Update` privilege for the attached database. Although you can access many databases during an RdbALTER session, you can attach to only one database at any one time.

If you specify a database root file (.rdb) in the RMU Alter command line, that database is attached as part of RdbALTER startup. In this case, a separate `Attach` command is unnecessary. Otherwise, no commands changing database pages are allowed until an `Attach` command naming that database is issued.

6.2.2 Clearing a Corruption Flag

Note

Oracle Corporation recommends that you discontinue using the RdbALTER `Uncorrupt` command. The `RMU Set Corrupt_Pages Consistent` command replaces the RdbALTER `Uncorrupt` command. The `RMU Set Corrupt_Pages Consistent` command makes a page, area, or all areas on a disk consistent.

When a storage area's corruption indication flag has been triggered as a result of a batch-update transaction failure, you need to reset the corruption flag manually. Use the `Uncorrupt` command to reset the storage area's corruption indication flag (`Filid Corrupt_Flg`). Then you can access the uncorrupted sections of a corrupted storage area. The format for the `Uncorrupt` command follows:

`Uncorrupt storage-area-name`

You can specify a storage area of the current database either by the storage area name (the name given by the `Area` clause in the database) or by the storage area number (assigned when the database is created and shown on the first line of a page display).

Storage areas are most often corrupted if rollbacks cannot occur and updates cannot be undone. Because there is no .ruj file for a batch-update transaction, the update cannot be undone, which sets the corruption flag.

The `Uncorrupt` command lets you access a database that is in an uncertain condition. Accordingly, you see the following message when you enter the `Uncorrupt` command for a corrupt database (for example from a failed batch-update transaction):

```
RdbALTER> UNCORRUPT 1

      ***** WARNING! *****

      BEWARE ATTEMPTING TO UNCORRUPT A STORAGE AREA
      WITHOUT FIRST VERIFYING IT USING THE RMU/VERIFY
      COMMAND.

      AN RdbALTER ROLLBACK COMMAND WILL LEAVE THIS
      AREA MARKED CORRUPT.
```

```
Area RDB$SYSTEM now marked uncorrupt.
```

6.2.3 Selecting the Area Page for Altering

When you attach to a database, `RdbALTER` automatically makes storage area 1 the current storage area and page 1 of that storage area the current page. To work on any other storage area and page, use the `Area . . . Page` command.

If you specify `Area` but not `Page`, `RdbALTER` makes the storage area you specify current and fetches page 1 of that storage area. If you specify both `Area` and `Page`, `RdbALTER` makes the storage area you specify current and fetches the page you specify from the new current storage area. You can specify a storage area of the current database by the storage area name (the name specified in the `Area` clause in the database):

```
RdbALTER> AREA EMPIDS_LOW
```

You can also specify a storage area by the storage area number (assigned when the database is created and shown on the first line of a page display):

```
RdbALTER> AREA 2
```

If you specify a storage area that does not exist, you receive an error message. When requesting a storage area, you can also specify the page to be altered. Express the page as a decimal integer from 1 to the number of pages in the storage area. For example, to select page 100 of storage area 3 use the following command:

```
RdbALTER> AREA 3 PAGE 100
```


RdbALTER can access only one page of one storage area at a time. You can fetch a page from the current storage area with the Page command. The format for the Page command follows:

```
Page [page-number]
```

The page number identifies a page to be altered in the current database storage area. If you issue Page without a page number, RdbALTER fetches the next page. If you issue Page without a page number and you are already at the highest numbered page of the storage area, RdbALTER fetches page one. If you specify a storage area or page that does not exist, you receive an error message.

You can use the RMU Dump command to obtain a list of the storage area, page, and record numbers for the database. For example, to obtain a list for the mf_personnel database, enter the following command:

```
$ RMU/DUMP/OPTIONS=DEBUG/OUTPUT=MFPERS_DUMP.LIS MF_PERSONNEL
```

This RMU Dump command produces a database list that includes storage area ID numbers for the mf_personnel database, and it writes the list to a file called mf_pers_dump.lis. For example, the EMPIDS_LOW storage area has an area ID of 2.

The contents of the output file are similar to the output you receive with the RMU Dump Header command with the exception that internal information about the data is not displayed with the RMU Dump command. This information displays with the Options=Debug qualifier and is useful for diagnostic support.

You can also use the RMU Dump command to obtain a display of the contents of selected database storage and logical areas that includes logical area ID numbers and record types. The storage area display helps you determine which records are stored on which pages. For example, to see a display of the EMPIDS_LOW storage area of the mf_personnel database, use the following command:

```
$ RMU/DUMP/AREAS=EMPIDS_LOW /START=1/END=2 -  
_$_ /OUTPUT=EMPIDS_LOW_DUMP.LIS MF_PERSONNEL
```

This RMU Dump command writes the EMPIDS_LOW area of the mf_personnel database to a file named EMPIDS_LOW_DUMP.LIS. The contents of the file EMPIDS_LOW_DUMP.LIS are shown in Example 6-1.

Example 6-1 RMU Dump Command to Display Contents of the EMPIDS_LOW Storage Area

```
$ RMU/DUMP/AREAS=EMPIDS_LOW /START=1 /END=2 /OUTPUT=EMPIDS_LOW_DUMP.LIS -
_$ MF_PERSONNEL
```

```
.
.
00000000000000000000000000000000 0090 free space '.....'
      :::: (2 duplicate lines)
00000000000000000000000000000000 00C0 free space '.....'
      001D 00CE line 14: record type 29
00 0001 00D0 Control information
      .... 38 bytes of static data
0B46C000474E454D3039313030000124 00D3 data '$..00190MENG.ÀF.'
474D5250008696563C2A800000839052 00E3 data 'R.....*<V...PRMG'
      C03931333030 00F3 data '00319À'
      00 00F9 padding '.'
.
.
      0042 2005 02AE line 5: bucket for hash index 66
      .... total hash bucket size: 51
FFFF FFFFFFFF FFFF 02B2 bucket overflow -1:-1:-1
      00 02BA flags 0
      00000004 02BB duplicate count 4
FFBE 00000002 0007 02BF duplicate node 66:2:7
      06 02C7 key len: 6 bytes
      353631303000 02C8 key: '.00165'
      00000004 02CE duplicate count 4
FFBE 00000002 000C 02D2 duplicate node 66:2:12
      06 02DA key len: 6 bytes
      303931303000 02DB key: '.00190'
      35 02E1 padding '5'
.
.
      003A 2005 035C line 2: bucket for hash index 58
      .... total hash bucket size: 51
FFFF FFFFFFFF FFFF 0360 bucket overflow -1:-1:-1
      00 0368 flags 0
      00000001 0369 duplicate count 1
003F 00000002 0001 036D pointer 63:2:1
      06 0375 key len: 6 bytes
      353631303000 0376 key: '.00165'
      00000001 037C duplicate count 1
003F 00000002 0003 0380 pointer 63:2:3
      06 0388 key len: 6 bytes
      303931303000 0389 key: '.00190'
      35 038F padding '5'
```

(continued on next page)

Example 6–1 (Cont.) RMU Dump Command to Display Contents of the EMPIDS_LOW Storage Area

```

001A 0390 line 1: record type 26
00 0001 0392 Control information
      .... 71 bytes of static data
0420886874696D53353631303000010B 0395 data '...00165Smith. .'
6E655420303231440D20847972726554 03A5 data 'Terry. .D120 Ten'
75726F636F684307209F2E7244207962 03B5 data 'by Dr.. .Chocoru'
4932C0004D3731383330484E12208B61 03C5 data 'a. .NH03817M.À2I'
00F032006B0E77 03D5 data 'w.k.2.'

2001 03DC line 0: SYSTEM record
02 000E 03DE 14 bytes in 2 sets/dynamic items
003A 06 03E1 6 bytes, storage set type 58
      12 2E 03E4 next 58:2:2
      00 03E6 owner 57:2:0
0042 07 03E7 7 bytes, storage set type 66
0095 5B 03EA next 66:2:5
      00 03ED owner 57:2:0

FFFFFFF 03EE snap page pointer -1
00000028 03F2 snap pointer TSN 40
000000000000000000000000 03F6 MBZ '.....'
00000000 03F8 page sequence number 0
00000000 03FC MBZ '....'
.
.
.

```

Example 6–1 shows selected parts of pages 1 and 2 of the EMPIDS_LOW storage area of the sample mf_personnel database. Page 1 is a SPAM page. When you display a data page such as page 2, you find the following:

- Oracle Rdb assigns ID numbers to logical areas. Logical areas are assigned to tables and indexes.
- Oracle Rdb assigns rows record-type identifiers according to the order in which tables are created in the database.

For example, page 2 contains logical area IDs 58 (JOB_HISTORY_HASH hashed index), 50 (EMPLOYEES_HASH hashed index), 55 (EMPLOYEES table), and 49 (System record), and the following record types, record type 25 (JOB_HISTORY rows) and record type 22 (EMPLOYEES rows).

Use the Area . . . Page commands in the RdbALTER environment to follow dbkey pointers. To display the first record on page 2 of the EMPIDS_LOW storage area (from line 1 on page 2 of area 2), enter the commands shown in Example 6–2.

Example 6–2 Display of Line 1 of Page 2 in Area 2 (EMPIDS_LOW Storage Area)

```
$ SET PROCESS/PRIVS=SYSPRV
$ RMU/ALTER MF_PERSONNEL
%RMU-I-ATTACH, now altering database "$DUAL:[ORION]MF_PERSONNEL.RDB:1"

RdbALTER> AREA 2 PAGE 2
RdbALTER> DISPLAY LINE 1

          001A 0390 line 1: record type 26
          00 0001 0392 1 byte in 0 sets/dynamic items
          .... 71 bytes of static data
0420886874696D53353631303000010B 0395 data '...00165Smith. .'
6E655420303231440D20847972726554 03A5 data 'Terry. .D120 Ten'
75726F636F684307209F2E7244207962 03B5 data 'by Dr.. .Chocoru'
4932C0004D3731383330484E12208B61 03C5 data 'a. .NH03817M.À2I'
          00F032006B0E77 03D5 data 'w.k.2.'
```

6.2.4 Displaying Page Contents

The Display command allows you to see the entire contents of a database page or any part of a database page with a Display command. The format for the Display command follows:

```
Display [ page-number ] [ option ]
```

See the *Oracle RMU Reference Manual* for complete Display command syntax. The page-number parameter identifies the page whose information you want to display. The current page is the default. An individual Display command can include only one of the following options:

- Asterisk (*)
Displays the entire page.
- Header
Displays the entire page header.
- The parameter page-number
Displays the 4-byte page number field.
- Storage-Area
Displays the 2-byte storage area identification.
- Checksum
Displays the 4-byte page checksum field.
- Time_Stamp
Displays the 8-byte timestamp field.

- **Free_Space**
Displays the 2-byte field indicating how much free space remains on the page.
- **Locked_Free_Space**
Displays the 2-byte field indicating how much free space is allocated for exclusive use by a recovery-unit journal.
- **COUNT**
Displays the 2-byte field showing the number of line index entries. If this number is 1, the page contains only the SYSTEM record.

Note

In the following two options, the integers denoting Index and Line are zero-based. For example, Index 0 refers to the first index, and Line 3 refers to the fourth line. The integer *n* is optional. The present value of the relevant pointer is the default. References to Index and Line are invalid if the current page is a SPAM page.

- **Index *n***
Displays the offset field, the length field, or both, from the line index indicated by *n*. For example, if you enter Display Index 3 Offset, the offset address field from the fourth line index displays.
- **Line *n***
Displays information from an individual storage segment. You can display the record-type field or the entire content of the storage segment line indicated by *n*.
- **Data *offset***
Displays 1, 2, or 4 bytes of data in the radix specified. If you do not specify a radix, the default radix is used. See the description of the Radix command in Section 6.2.10 for information on how to set a default radix.
- **Space *range***
Displays a specified range of SPAM entries; it is valid only if the current page is a SPAM page. The range value can be an asterisk (*), referring to all entries or a set of consecutive entries that you describe as follows:

lower-data-page-number[:higher-data-page-number]

Each entry on a SPAM page consists of 2 bits, containing a value 0 through 3 that represents a fullness threshold. For example, if the *n*th SPAM entry contains a 2, it means that the *n*th data storage page in the interval has reached a percentage of fullness greater than the second threshold for the area, but less than or equal to the third threshold.

You can use the Display command with its option to see a variety of information on a database page. For example, Example 6–2 shows how to fetch page 2 of storage area 2 and display the first line on the page. Page 2 remains the current page until you specify a different page number. Likewise, line 1 remains the current line until you specify a different line number.

6.2.5 Changing Page Contents

You can use the Deposit command to alter specified fields on the current database page. You would usually use RdbALTER to correct a bad checksum or change dbkey pointers. RdbALTER makes changes in virtual memory. The changes are not actually written to the database until you enter a Commit command.

Deposit is the default command at RdbALTER command level; if no command follows the RdbALTER> prompt, RdbALTER automatically parses the command line as Deposit. The general format for the Deposit command follows:

```
[Deposit] { option = ascii-string }
```

Note

Do not issue a Deposit command immediately after a Rollback command because a warning message will be returned indicating that there is no current page. The Rollback command removes the current page context. For this reason, you must specify your location again by resetting the RdbALTER current page context, using either a Display, Page, or Area command before you specify the Deposit command.

The syntax of most Deposit command options is the same as the Display command option syntax. See the *Oracle RCU Reference Manual* for a complete description of the Deposit command syntax.

The ASCII-string parameter specifies the new value of the field you are patching. The value is deposited in ASCII as entered, unless you request otherwise in one of the following ways:

- By specifying Hexadecimal or Decimal in a prior Radix command

- By specifying Hexadecimal or Decimal in a Deposit Data command
- By enclosing ASCII data in quotation marks (" ")

Timestamps and file specifications (including the root file on attach) must always be enclosed in quotation marks because they include punctuation characters. Specify the field to be patched just as you would specify a field to be displayed. See the Display command description in Section 6.2.4 for further information on field specification.

The patch field specification must be followed by an equal sign (=) and a string of characters specifying the new value of the patched field.

Whenever you alter something on a page, that page's checksum changes. To repair some bad data or a dbkey pointer, for instance, you must perform the following tasks:

1. Deposit the correct data or the correct dbkey pointer.
2. Verify the page to make sure its format is correct and to get the new checksum.
3. Deposit the correct checksum.
4. Verify the changes.
5. Commit the changes.

The following procedure demonstrates how to recover from the corruption of data on a database page using RdbALTER:

1. Verify the data page to get the new checksum.
2. Deposit the new checksum value on the data page in order to change the erroneous checksum.
3. Verify the data page again.

To corrupt a database page and to fix the corruption, do the following:

1. Verify the database with the RMU Verify All command to be certain the database is not corrupt:

```
$ RMU/VERIFY/ALL/LOG MF_PERSONNEL
```

The absence of warning messages indicates that the database is not corrupt.

2. Corrupt the page by using the Deposit command to change Terry Smith's name to Jerry Smith in area 2 (EMPIDS_LOW), page 2, line 1, as shown in Example 6–3. You must set the Radix to Hexadecimal to enter the values in hexadecimal.

Example 6–3 Changing Data on Page 2 in the EMPIDS_LOW Storage Area

```
$ RMU/ALTER MF_PERSONNEL
%RMU-I-ATTACH, now altering database "$DUAL:[ORION]MF_PERSONNEL.RDB;1"
```

```
RdbALTER> AREA 2 PAGE 2
RdbALTER> DISP LINE 1
```

```
          001A 0390 line 1: record type 26
          00 0001 0392 1 byte in 0 sets/dynamic items
                .... 71 bytes of static data
0420886874696D53353631303000010B 0395 data '...00165Smith. .'
6E655420303231440D20847972726554 03A5 data 'Terry. .D120 Ten'
75726F636F684307209F2E7244207962 03B5 data 'by Dr.. .Chocoru'
4932C0004D3731383330484E12208B61 03C5 data 'a. .NH03817M.À2I'
          00F032006B0E77 03D5 data 'w.k.2.'
```

```
RdbALTER> RADIX HEXADECIMAL
RdbALTER> DEPOSIT DATA /BYTE 03A5=4A
RdbALTER> DISP LINE 1
```

```
          001A 0390 line 1: record type 26
          00 0001 0392 1 byte in 0 sets/dynamic items
                .... 71 bytes of static data
0420886874696D53353631303000010B 0395 data '...00165Smith. .'
6E655420303231440D2084797272654A 03A5 data 'Jerry. .D120 Ten'
75726F636F684307209F2E7244207962 03B5 data 'by Dr.. .Chocoru'
4932C0004D3731383330484E12208B61 03C5 data 'a. .NH03817M.À2I'
          00F032006B0E77 03D5 data 'w.k.2.'
```

3. Verify the page, using RdbALTER to determine the new checksum value shown in Example 6–4.

Example 6–4 Page Verification Using the RdbALTER Utility Returns a Checksum Error

```
RdbALTER> VERIFY
%RMU-W-PAGCKSBAD, area EMPIDS_LOW, page 2
                contains an invalid checksum
                expected: 7876961C, found: 7876A01C
```

4. Using RdbALTER, deposit the expected checksum value, 192892E5.
5. Display the header for page 2 to confirm the change.
6. Verify page 2 again, as shown in Example 6–5.

Note that checksums must be expressed in Hexadecimal, so you must first use the Radix command to indicate Hexadecimal radix.

Example 6–5 Depositing a New Checksum on Page 2 in the EMPIDS_LOW Storage Area

```
RdbALTER> DEPOSIT CHECKSUM = 7876961C
RdbALTER> DISPLAY HEADER
      0002 00000002  0000  page 2, area 2
      7876961C  0006  checksum = 7876961C
009723ED 1EC8B380  000A  time stamp = 8-SEP-1993 16:24:41.40
      0000 003E  0012  62 free bytes, 0 locked
      000F  0016  15 lines

RdbALTER> VERIFY
RdbALTER>
```

Because no error message displays, the page verification is successful. At this point, no changes have been written to the database page until you commit the change by using the RdbALTER Commit command. You also have the option of discarding these changes with a Rollback command.

In a more realistic situation, you might receive results from an RMU Verify command that indicate corruption in area EMPIDS_LOW (2) due to an invalid checksum. From these results, the RMU Verify command indicates what the checksum should be, as well as what it currently is. Then you enter RdbALTER and use the Attach command to attach to the mf_personnel database and investigate the problem by inspecting the contents of the page.

7. Enter the Commit command, as follows:

```
RdbALTER> Commit
```

8. Exit from RdbALTER and check your work with the RMU Verify command, as follows:

```
RdbALTER> Exit
$ RMU/VERIFY/ALL/LOG MF_PERSONNEL
```

If no error messages display from the full verify operation, then your changes are sound and the database is not corrupt.

6.2.6 Moving Database Files

If you are expanding the system from one node to a cluster, or if a storage area has grown too large for its current storage device, or if you want to move database files, use one of the RMU utilities. Oracle Rdb recommends that you either:

- Use the RMU Move_Area command to relocate storage area files or use the RMU Copy_Database command to perform total database relocation.

Note

You must perform a full backup operation immediately after you perform an RMU Move_Area or RMU Copy_Database operation. This is because, if parameters are changed during the move or copy operation, the restore and recover operations might not be able to recreate the database correctly.

Neither command requires intermediate media nor the restoration of the .rbf file, and for either command, the operation is performed more quickly than any alternative operation.

- Use the RMU Backup and RMU Restore commands to relocate database files, but this requires intermediate media or a restore operation of the .rbf file.
- If the database root file (.rdb) needs to be updated for some reason, consider using the RMU Restore Only_Root command.

You should consider using the Deposit command only as a last resort and only if you cannot use an existing RMU command (such as RMU Move_Area, RMU Copy_Database, RMU Backup and RMU Restore, and RMU Backup and RMU Restore Only_Root) to do the task you have planned. For example, if you need to change a system-concealed logical name or change a reference to a disk drive in the root file because the name is no longer correct due to a disk drive problem, you can use the Deposit command. However, use the RMU Restore Only_Root command in either case.

See Chapter 8 and the *Oracle RMU Reference Manual* for more information on using these commands. See Section 6.2.8 for another example of using the Deposit Root command.

Note

Exercise caution when working with multfile databases. All modified file specifications in the root file must exactly match all new database file locations for the database to be usable.

6.2.7 Moving Data

The following syntax shows how you can use the Move command to move a string of data to a different location on the page:

Move old-offset-start : old-offset-end new-offset

The *old-offset-start* and *old-offset-end* parameters specify the hexadecimal offset addresses of the first and last bytes in the data sequence to be moved. The *new-offset* parameter specifies the hexadecimal offset address of the first byte in the sequence of bytes that receives the moved data.

The receiving field, defined by the *new-offset* parameter and the length of the sending field, is replaced by the contents of the sending field. The number of bytes moved is presented as follows:

$(old-offset-end) - (old-offset-start) + 1$

The sending field and all other information in the page remain the same.

Offset addresses are discussed more fully in Chapter 11.

The following procedure provides an example of using the Move command for a situation that may rarely happen. In fact, with problems such as this, the recommended action for checksum errors is to restore and recover the page that contains an invalid checksum. This example is only for illustration of the Move command. In Example 6-6, a database page is verified and returns an invalid checksum for a database page.

Example 6–6 Checksum Error Returned from a Full Verification

```
$ RMU/VERIFY/ALL MF_PERSONNEL
%RDB-F-IO_ERROR, input or output error
-RDMS-F-CANTREADDBS, error reading pages 8:3-3
-RDMS-F-CHECKSUM, checksum error - computed 967AC1E7, page contained 1B4DA010
%RMU-W-PAGCKSBAD, area EMP_INFO, page 3
                    contains an invalid checksum
                    expected: 967AC1E7, found: 1B4DA010
%RMU-E-CORRUPTPG, Page 3 in area EMP_INFO is marked as corrupt.
```

An inspection of the checksum error message shows that the problem occurs in area 8 (EMP_INFO) on page 3 and indicates that data on the database page has changed in some way. A more detailed view of the data page (Example 6–7) shows that there are **** overlap **** error messages within line entry 5. This message usually indicates that data is no longer found where the line index offset address indicates that it should be.

Use the following steps to inspect the contents of page 3:

1. Check the line index entries and note that line 5 starts at HEX 00EE and is 30 bytes long.
2. Check the actual record for entry 5.
3. Perform the following series of checks on entry 5:
 - a. Note that the record type indicates 256, but the record is a DEGREES record.
 - b. On page 3, line 21, you determine that a DEGREES record is a record type 32 (HEX 20).
 - c. The actual record length is 25 bytes plus 5 bytes of overhead for a total of 30 bytes, which corresponds with the line index entry of 30 bytes. However, information is missing, such as the number of bytes of static data. Also, extra information displays such as the number of bytes, storage set type 12288, next record, and owner record.
 - d. One byte, HEX 20, is in free space where no bytes should be displayed. The value HEX 20 is decimal 32. This seems to be the actual record type identifier, but offset by 1 byte.
 - e. Check to see if the complete record exists. The record appears to be intact except that the last byte is repeated twice, so the record appears to be offset by 1 byte into free space. This means the record must be moved back to the real starting offset address indicated in line index 5 as HEX 00EE.

Example 6-7 Display of Page 3 of the EMP_INFO Storage Area

```
$ RMU/ALTER MF_PERSONNEL
%RMU-I-ATTACH, now altering database "DUAL2:[DB]MF_PERSONNEL.RDB;1"

RdbALTER> AREA 8 PAGE 3
RdbALTER> RADIX HEXADECIMAL
      0008 00000003 0000 page 3, physical area 8
      1B4DA010 0006 checksum = 1B4DA010
009723ED 1EC8B380 000A time stamp = 8-SEP-1993 16:24:41.40
      0000 0026 0012 38 free bytes, 0 locked
      0016 0016 22 lines
.
.
.
      001E 00EE 002C line 5: offset 00EE, 30 bytes
.
.
.
      001E 010C 006C line 21: offset 010C, 30 bytes
.
.
.
005000500050005000500050005000500050 00C8 free space 'P.P.P.P.P.P.P.P.'
000000500050005000500050005000500050 00D8 free space 'P.P.P.P.P.P.P.P...'
      200000000000 00E8 free space '..... '
      0100 00EE line 5: record type 256
      13 0000 00F0 Control information
      3000 01 00F3 1 byte, storage set type 12288
      3631 30 00F6 next 11:3:1590
      544142 35 00F9 owner 12:7:327
      ===== ** overlap **
      3030 00 00F4 0 bytes, storage set type 12336
      423536 31 00F7 next 10:3:13627
      B44554 41 00FB owner 12:17751:5
      ===== ** overlap **
.
.
.
```

(continued on next page)

Example 6–7 (Cont.) Display of Page 3 of the EMP_INFO Storage Area

```

      3030 00 00F4 0 bytes, storage set type 12336
      423536 31 00F7 next 10:3:13627
      B44554 41 00FB owner 12:17751:5
      ===== ** overlap **
      ===== ** overlap **

      00E0 010C line 21: record type 224
      00 0001 010E Control information
      .... 25 bytes of static data
414207BA44574F423537313030000113 0111 data '...00175BOWD°.BA'
      E000208A7374724120 0121 data ' Arts. .à'
      .
      .
      .

      001F 03A8 line 1: record type 31
      00 0001 03AA Control information
      .... 54 bytes of static data
756F5320666F202E5541435355000129 03AD data ')..USCAU. of Sou'
61696E726F66696C6143206E72656874 03BD data 'thern California'
414307208873656C65676E4120736F4C 03CD data 'Los Angeles. .CA'
      E03733303239 03DD data '92037à'
      00 03E3 padding '.'

      2001 03E4 line 0: SYSTEM record
      00 0001 03E6 1 byte in 0 sets/dynamic items
0000000000 03E9 padding '.....'

      00000001 03EE snap page pointer 1
      00000220 03F2 snap pointer TSN 544
      0000 03F6 MBZ '..'
      00000000 03F8 page sequence number 0
      00000000 03FC MBZ '.....'

```

4. Move record entry 5 back to where it is supposed to be.
5. Clean up any extraneous bytes on the database page.

Notice that free space should display all zeros, but every other byte contains HEX 50 or decimal P. These bytes need to be reset to zeros. Use the RdbALTER Deposit Data Byte command to modify this data by indicating the address and value 00 to make the change. See Example 6–8 for an example.

6. Verify the page as shown in Example 6–8.

Example 6–8 Moving Contents of Line 5 to a Designated Location, Cleaning Up Extraneous Bytes, and Verifying Storage Pages

```
RdbALTER> MOVE 00ED:010B 00EE
RdbALTER> DEPOSIT DATA /BYTE 00ED=00
RdbALTER> VERIFY
%RMU-W-PAGCKSBAD, area EMP_INFO, page 3
                contains an invalid checksum
                expected: 191D9D90, found: 1B4DC010
RdbALTER> DEPOSIT CHECKSUM =191D9D90
RdbALTER> VERIFY
RdbALTER>
```

If page verification indicates a page checksum error and you know the data is correct and the remainder of the page contents are correct, reset the page checksum to its expected value by using the Deposit Checksum command as shown. In this example, because the page checksum does not match exactly what it should be, some information still remains to be corrected, but that information is not clearly apparent. This checksum discrepancy is an excellent example of why Oracle Rdb recommends you restore and recover database pages rather than use the RdbALTER utility to patch data pages to avoid uncertainty.

7. If the data page verifies without errors, use the Commit command to commit the changes.
8. Use the Exit command to exit from the RdbALTER utility.
9. Perform a full verification or at least a verification of the EMP_INFO storage area, as shown in Example 6–9.

Example 6–9 Committing Changes, Exiting from RdbALTER, and Performing an RMU Verify operation of the EMP_INFO Storage Area of mf_personnel

```
RdbALTER> COMMIT
RdbALTER> EXIT
$ RMU/VERIFY/AREAS=EMP_INFO MF_PERSONNEL
%RMU-E-CORRUPTPG, Page 3 in area EMP_INFO is marked as corrupt.
$
```

10. Because page 3 is still marked as corrupt, you must reset its consistent flag by using the RMU Set Corrupt_Page command. You must be certain that the contents of this page are accurate before proceeding. That is, are the contents what they should be. You should inspect the page carefully

looking for problems. Seeing none, proceed to make the page consistent again.

```
$ RMU/SET CORRUPT_PAGES /CONSISTENT /AREA =8 MF_PERSONNEL
```

```
***** WARNING! *****
```

```
Marking a storage area or page consistent does not
remove the inconsistencies. Remove any inconsistencies
or corruptions before you proceed with this action.
```

```
Do you wish to continue? [N] Y
```

```
%RMU-I-AREAMARKED, Area 8 was marked consistent.
```

To check that this page is no longer corrupt, show the corrupt page table (CPT) by using the RMU Show Corrupt_Pages command as follows:

```
$ RMU/SHOW CORRUPT_PAGES MF_PERSONNEL
```

```
*-----*
* Oracle Rdb V6.0-0                               16-SEP-1993 16:50:23.18
*
* Dump of Corrupt Page Table
* Database: $DUAL:[ARTIE]MF_PERSONNEL.RDB;1
*
*-----*
```

```
Corrupt page table is empty.
```

```
$
```

11. When database verification returns no errors, the database is once again ready for use.

6.2.8 Using the RMU Alter Command to Remove References to .ruj Files

The RMU Alter command allows a user to remove references to .ruj files from the root file if one or more .ruj files are accidentally deleted; thus, allowing access to the database. However, this action marks the database as “eternally corrupt” which results in a warning message in all future system management (RMU) functions against this database. The database is either structurally corrupt, logically corrupt (violates a constraint), or “user data” corrupt (balance is \$250.00 instead of \$300.00).

Using these commands to make changes permits users to attach to the database with the realization that the database is corrupt in situations where the .ruj files have been accidentally deleted. The database must be restored and recovered from clean backup files in order to guarantee the consistency of its contents.

The following RdbALTER utility syntax supports removal of .ruj file references from the root file if any .ruj files are deleted:

```
Display Root User n RUJ_FILENAME
```

or

```
[Deposit] Root User n RUJ_FILENAME = filename
```

In the syntax examples, the variable file name can be " ".

You must specify a version number with the .ruj file name when you use this RdbALTER utility.

To correct an “eternally corrupt” database, use any of the following methods:

- Restore and recover from a clean backup and .aij file.
- Export and import the database (this could result in errors).
- Unload the database, re-create the database, and load the database (this could result in data access errors).

Only the first option (restore and recover) ensures that all types of corruption are actually handled. The other methods may leave “user data” corruption. The proper timing for changing the .aij file relative to an online incremental or full backup operation is explained in detail in Section 9.6.1. The basic procedure follows:

1. Close the database and restrict access.
2. Change or switch the .aij file or back up the .aij file.
3. Open the database.
4. Back up the database and allow processing to continue during the backup operation.

6.2.9 Clearing an Inconsistent Flag

Note

Oracle Corporation recommends that you discontinue using the RdbALTER Uncorrupt command. The RMU Set Corrupt_Pages Consistent command replaces the RdbALTER Uncorrupt command. The RMU Set Corrupt_Pages Consistent command makes a page, area, or all areas on a disk consistent.

When a storage area is restored from a backup file on a per-area basis, it does not reflect data that has been updated since the backup operation. The transaction level of the restored storage area reflects the transaction level of the backup file, not the transaction level of the database. Therefore, the transaction level of the restored storage area differs from that of the database. Oracle Rdb marks the storage area by setting a flag to “inconsistent” in the storage area file.

When a storage area’s inconsistent indication flag has been triggered, you need to do an AIJ recovery for that storage area. Only if an AIJ recovery is not possible because the .aij file is lost, was accidentally deleted, or contains AIJ errors, and you are willing to accept the possible corruption to the database because recovery is not possible, should you manually reset the inconsistent flag. Under these circumstances you can use the Make Consistent command to reset the storage area’s inconsistent indication flag. The format for the Make Consistent command follows:

```
Make Consistent storage-area-name
```

You can specify a storage area of the current database either by the storage area name (the name given by the Area clause in the schema) or by the storage area number (assigned when the database is created and shown on the first line of a page display).

You can perform a recovery, by storage area, to upgrade the transaction level of the restored storage area to that of the database. (After-image journaling must be enabled in order to restore by storage area.) If you are sure that no updates have been made to the database since the backup operation, you can use the Make Consistent command in RdbALTER to change the setting of the flag from inconsistent to consistent.

The Make Consistent command lets you access a database. Accordingly, the following message is displayed when you enter the Make Consistent command:

```
***** WARNING! *****  
  
BEWARE ATTEMPTING TO MAKE CONSISTENT A STORAGE  
AREA WITHOUT FIRST VERIFYING IT USING THE  
RMU/VERIFY COMMAND.  
  
AN RdbALTER ROLLBACK COMMAND WILL LEAVE THIS  
AREA MARKED INCONSISTENT.
```

6.2.10 Changing the Radix

The Radix command sets the default radix for entering numeric data to hexadecimal or decimal. The format for the Radix command follows:

```
Radix { option }
```

The Radix command option must be either Decimal *or* Hexadecimal format. Depending on which option you select, you can then enter numeric data in either Decimal or Hexadecimal.

This command does not change the radix for specifying page offsets. Page offsets must always be specified in hexadecimal radix.

6.2.11 Verifying Alterations

You can use the Verify command as a preliminary diagnostic tool to perform the following tasks:

- Locate integrity problems.
- Check your alterations prior to issuing the Commit command that writes the changes to the page.

The Verify command performs a static verification of the page header and the page checksum. The format for the Verify command follows:

```
Verify
```

A static verification performs only a page header and page checksum verification of the database page. The RdbALTER Verify command is similar to an RMU Verify Areas=* Checksum_Only command, except the RdbALTER Verify command checks only the current page and verifies the page header.

If the database uses space management and the current page is a data storage page, RdbALTER Verify checks the SPAM entry for the current page to see that it contains the correct value. If the current page is a SPAM page, the Verify command checks the page for valid syntax but does not check the individual SPAM entries for correctness. If the page is corrupt, error messages display.

6.2.12 Keeping a Log or an Audit Trail of Alterations

You can keep a log or an audit trail of all or part of an RdbALTER session by entering a Log command. Discontinue the audit trail by entering a Nolog command. The Log command initiates an audit trail of all terminal input and output and writes the audit information to the file you specify. The format for the Log command follows:

```
Log file-spec
```

You can specify the file that contains the audit trail log by using the file specification parameter. The default file type is .lis. For example, to send the audit information to a file named audit.lis, enter the following command:

```
RdbALTER> Log audit
```

The following command sends the audit information to the file audit.trl:

```
RdbALTER> Log Audit.trl
```

The Nolog command stops RdbALTER logging. To stop sending the audit information to the log file, use the Nolog command as follows:

```
RdbALTER> Nolog
```

The Nolog command stops audit trail logging if a previous Log command is still in effect. The Exit command performs an implicit Nolog operation if the Log command is still active; thus, you need not enter a Nolog command before exiting the RdbALTER session. You cannot exit until you enter a Commit or a Rollback command.

6.2.13 Completing Transactions

RdbALTER stores altered pages in virtual memory until you issue a Commit or a Rollback command. This feature allows you to perform trial page alterations without necessarily writing them to the database. You should use the RdbALTER Verify command to check structural integrity of the altered page before committing changes to the database.

When you issue the Commit command, RdbALTER writes the changes into the database. You can alter any number of pages between Commit commands. The format for the Commit command follows:

```
Commit
```

Enter the Commit command to have all changed pages written to the database in their new form. If you believe you have made the problem worse or decide you cannot solve the problem with RdbALTER, enter a Rollback command. This command cancels your alterations and does not write the changes to the database. The format for the Rollback command follows:

```
Rollback
```

Follow Rollback commands with a Display, an Area, or a Page command to reset the currency indicator to the desired location. You must issue either a Commit or a Rollback command before exiting RdbALTER.

6.2.14 Exiting from the RdbALTER Utility

When you are finished altering a database, enter the Detach command. This command releases the Exclusive Update lock for the database so other users can access it again. The format for the Detach command follows:

Detach

You cannot alter storage areas within the database unless you attach to it again.

After issuing the Detach command, you can issue an Exit command to return to operating system command-line level. The format for the Exit command follows:

Exit

After changing data, you cannot issue the Exit command from the RdbALTER utility until you issue either a Commit or a Rollback command. This procedure protects against accidentally changing a database instance without providing a way to recover. Normally, control is returned to the operating system. If you issue an Exit command, and any altered but uncommitted pages exist, a message tells you to issue either a Commit or a Rollback command. RdbALTER does not exit until the Commit or the Rollback operation has accounted for all altered pages. Exit performs an implicit Nolog command if a log file is open.

6.2.15 Accessing Online Information

To access online reference information on the syntax and functions of RdbALTER commands, use the Help command. Help provides information about RdbALTER commands, terminology, and concepts. The format for the Help command follows:

Help [*keyword ...*]

The keyword parameter specifies an RdbALTER keyword or concept.

The RdbALTER Help utility functions exactly as the OpenVMS Help Facility does. If you enter Help with no keyword, a brief description of RdbALTER with a list of the RdbALTER commands or keywords appears. If you enter Help followed by a command, a brief description of the command displays with a list of keywords you can type to obtain further information.

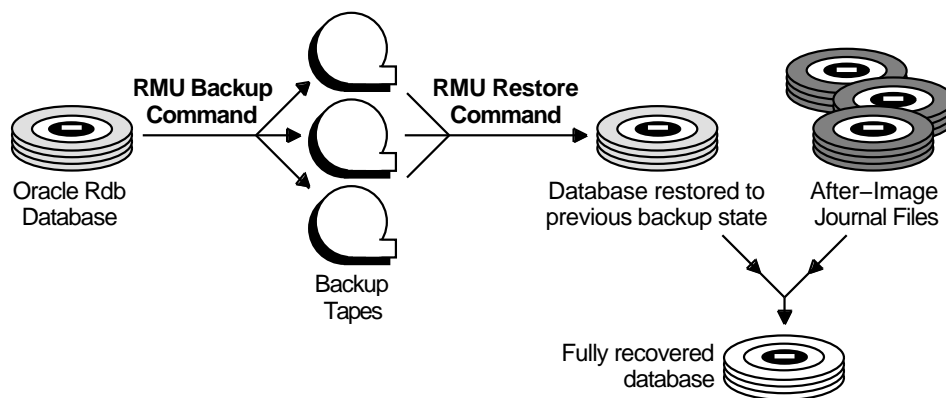
Backing Up Your Database

This chapter describes how you can safeguard your data against loss or corruption by creating backup copies of your database files using the Oracle RMAN backup utility.

7.1 Introduction to Database Backup

A database backup utility copies the files that comprise the database to a file located on a separate disk or tapes. If anything happens to your database and you have backed it up regularly, you can restore the database to the state it was in when you last backed it up. Furthermore, if you keep after-image journal (.aij) files for the database (described in Chapter 9), you can recover transactions made after the backup operations. Figure 7-1 shows the backup and restore operations.

Figure 7-1 Backup and Restore Operations



NU-3574A-RA

Using both database backups and after-image journaling, you can rebuild a lost or corrupt database to its most current state by restoring the database from the backup file and applying transactions saved in after-image journal files to the restored copy of the database.

7.2 How Does Oracle RCU Back Up a Database?

Traditionally, a backup operation is a single-threaded application that reads data from disk files and then writes the data to backup devices. These sequential applications are sufficient for backing up files or directories selectively, but often are insufficient for a database backup. Moreover, most backup utilities require that you close the database; there can be no users accessing the database during the backup process.

The Oracle RCU backup utility provides a sophisticated, multithreaded backup operation designed to handle the complexities involved in backing up a database. The Oracle RCU backup operation is referred to as **multithreaded** because the utility processes multiple I/O operations at the same time. The following table describes the default and optional methods of performing Oracle RCU backup operations:

Type of Backup Operation	Description
Default	Uses a single, multithreaded process that performs multiple I/O operations to back up the database. This backup operation is the default for an Oracle Rdb database and it is provided with the base Oracle Rdb software kit.
Parallel	Uses multiple, multithreaded processes that work in parallel to back up the database. Each process performs multiple parallel I/O operations simultaneously to back up the database. Parallel backup is optional software and is separately licensed.

Both methods of Oracle RCU backup operations can perform backup operations for databases that are on line and openly accessible to users. During an online backup operation, users can attach to the database and execute any type of transaction that does not conflict with the read-only online backup transactions.

Sections 7.2.1 and 7.2.2 describe the Oracle RCU backup operations in more detail.

7.2.1 Oracle RMAN Default Backup Operation

The Oracle RMAN backup operation uses a single process to perform multiple I/O operations; the process simultaneously performs read operations to the database and write operations to one or more tape devices.

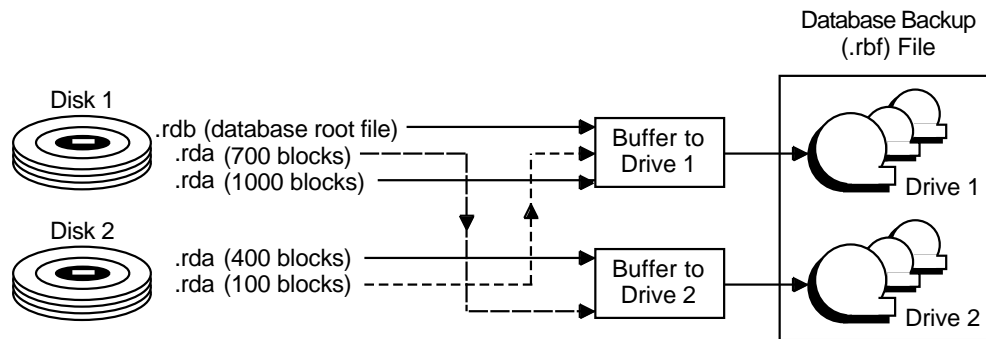
In a multithreaded backup operation, each storage area in a multifile database is assigned its own thread. In addition, if you have more than one master tape drive, the backup procedure assigns a backup thread to each drive.

Note

Multithreaded processing occurs only when you back up the database to tape devices. The Oracle RMAN backup operation is single threaded if you back up to a disk device.

For example, Figure 7-2 shows a database that has four storage areas and two master tape drives. The Oracle RMAN backup operation assigns threads to each of four storage areas and assigns backup threads to both tape drives.

Figure 7-2 Multithreaded Oracle RMAN Backup (Single Process)



NU-3577A-RA

The Oracle RMAN backup operation automatically balances the workload and I/O activity across all the storage areas and backup devices. Notice the following about the Oracle RMAN backup operation in Figure 7-2:

1. When the backup operation starts, the Oracle Rdb database root (.rdb) file is backed up first. The backup procedure always locates the database root file in its entirety on the first tape written on Drive 1.

Note

If the database root file is too large to fit on the first backup device, an error occurs. However, the error is unlikely to occur unless you write the backup file to a tape that already contains files. (For example, if you back up to a tape that has not been rewound (using the NoRewind qualifier).)

2. After Oracle RMU backs up the database root file, it backs up all the storage areas. Oracle RMU assigns storage areas to tape drives based on the number of database pages in each storage area. The backup procedure assigns the largest storage area to the first tape drive and the second largest storage area to the second tape drive. Remaining storage areas are assigned in an attempt to balance the number of blocks Oracle RMU writes to each buffer.

Oracle RMU displays messages telling you which backup threads are assigned to each drive and which storage area threads are assigned to each backup thread. You can see an example of this output in Example 7-2.

3. The tape backup thread writes the buffer (filled by the storage area threads) to tape. You use qualifiers on the RMU Backup command to specify the number of buffers the backup thread controls and allocates to the storage area threads. The number of buffers is equivalent to the maximum number of active write operations to the backup device.
4. If you have more than one master tape drive, the backup procedure writes to each tape drive independently.

7.2.2 Oracle RMU Parallel Backup Operation

OpenVMS OpenVMS
VAX Alpha

On OpenVMS systems, Oracle RMU can also perform a database backup using multiple multithreaded processes working in parallel. This option, called a **parallel backup**, uses the default Oracle RMU multithreaded backup process, but it allows you to distribute the work among multiple processes to improve load balancing and performance. (All Oracle RMU backup operations are “parallel” in the sense that the backup process is multithreaded. However, with the default Oracle RMU backup operation, only one thread can run at a time.)

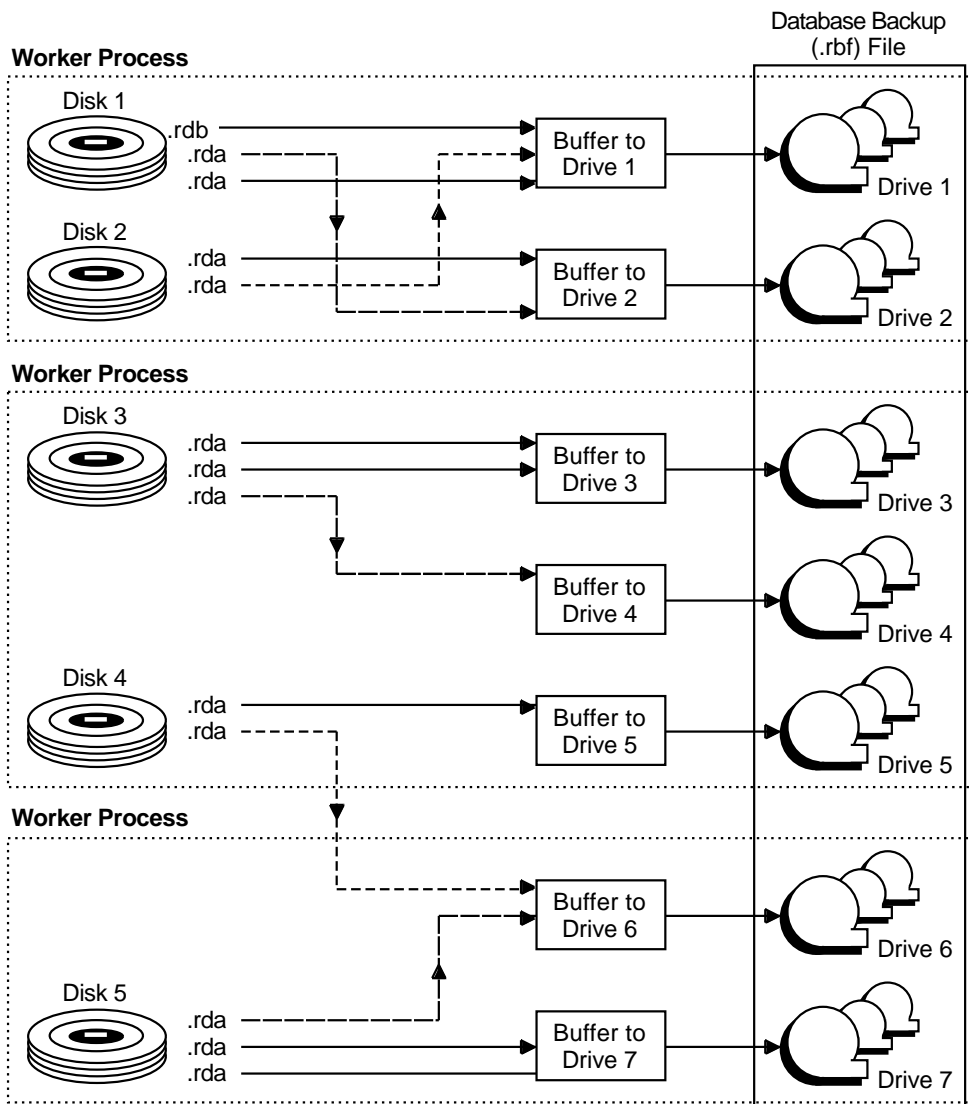
Note

You must have Oracle SQL/Services installed to perform a parallel backup operation. Refer to the *Oracle SQL/Services Installation Guide*

and the *Oracle SQL/Services Release Notes* for more information about installing and using the Oracle SQL/Services.

Figure 7-3 shows how a parallel backup operation uses multiple, multithreaded subprocesses called **worker processes** that back up assigned storage areas using assigned tape drives. By breaking the work load across multiple worker processes, you enhance the speed of the backup operation on OpenVMS systems.

Figure 7-3 Multithreaded Oracle RMAN Backup Using Multiple Parallel Processes



NU-3578A-RA

Each worker process can run on a different CPU on a symmetric multiprocessing (SMP) machine or on a different node in a cluster to further enhance the speed of the operation. The parallel backup operation is most beneficial when you use three or more processors in an SMP machine.

The worker processes are coordinated by a master **coordinator** process. The coordinator process serves as the backup operation execution manager. You specify information about the coordinator process and how the backup operation is split across worker processes in a **plan file**.

Section 7.5.6 and Section 7.10 describe parallel backup operations in more detail. ♦

7.3 Invoking the RMU Backup Command

When you invoke the RMU Backup command, Oracle RMU reads an Oracle Rdb database and writes the output to a single output file. On the RMU Backup command line, you must specify the database root file (.rdb) that you want to back up and where you want the backup operation to place the resulting backup (.rbf) file. The following example shows the Backup command syntax on a Digital UNIX system:

```
$ rmu -backup root-file.rdb backup-file.rbf
```

Note

Before you perform a backup operation in a clustered OpenVMS system, you must:

- Define the logical names SQL_USERNAME and SQL_PASSWORD
 - Provide proxy access (for the user that starts the backup operation) between all nodes involved in the backup operation
-

By default, Oracle RMU backs up all database pages in all storage areas of an Oracle Rdb database. Also, you can use qualifiers on the Backup command line to customize the backup operation. For example, you might back up only database pages in selected storage areas, or only the pages that have changed since the time of the last complete backup operation.

Oracle Corporation recommends using the RMU Backup command to protect both single-file and multfile databases. The Oracle RMU backup utility performs better than other backup mechanisms because Oracle RMU is designed specifically for use with Oracle Rdb databases.

Note

Do not use other backup utilities such as the OpenVMS Backup (BACKUP) utility or the Digital UNIX `tar` function to back up and restore Oracle Rdb databases. Relying on backup utilities other than Oracle RMU can compromise the reliability and availability of the database. See Section 7.4 for more information.

Although many discussions in this chapter include examples of how to use the RMU Backup command, you should also refer to the *Oracle RMU Reference Manual* for a complete description and examples.

7.3.1 What Does Oracle RMU Back Up?

The Oracle RMU backup operation provides a range of backup options that allow you to back up the entire database, individual storage areas, or only the database pages that have changed since the last full backup operation. There are four basic types of Oracle RMU backup operations:

- Full and complete
- Full and by area
- Incremental and complete
- Incremental and by area

Table 7–1 describes these Oracle RMU backup operations and the Backup command qualifiers you use to implement each type.

Table 7–1 Types of Oracle RMAN Backup Operations

Database Page Selection	Storage Area Selection	
	Complete (All Storage Areas)	By-Area (Selected Storage Areas)
Full (All database pages.)	<p>Copies the database root (.rdp) file and all the database pages in all the storage areas in the database. This is the default backup operation. You must use this type of backup prior to upgrading to a newer version of Oracle Rdb.</p> <p>Qualifiers: None required. This is the default backup operation.</p>	<p>Copies the database root (.rdp) file and backs up only the database pages in the storage areas that you specify on the RMAN Backup command line. All the storage areas are backed up only if you specify them all (or perform a full and complete backup operation).</p> <p>Qualifiers: Specify the storage areas using one of the following qualifiers:</p> <ul style="list-style-type: none"> • Include=[storage_area[...]] • Exclude=[storage_area[...]]
Incremental (Only pages that have changed since the last full and complete backup.)	<p>Copies the database root (.rdp) file and all database pages that have been updated since the latest full backup operation. Free space on those pages, and snapshot (.snp) file pages will not be copied to the backup file.</p> <p>Qualifiers: Use the following qualifier:</p> <ul style="list-style-type: none"> • Incremental (same as Incremental=Complete) 	<p>Copies the database (.rdp) file and only the database pages for the specified storage areas that have changed since the latest full backup operation.</p> <p>Qualifiers: Specify the storage areas using one of the following qualifiers on the RMAN Backup Incremental=By_Area command:</p> <ul style="list-style-type: none"> • Include=[storage_area[...]] • Exclude =[storage_area[...]]

For most databases, you might want to consider some combination of full and complete backup, and incremental backups. For example, you might want to perform a weekly full and complete backup, and nightly incremental backups.

Section 7.5 provides guidelines to help you determine a backup strategy.

7.3.2 Writing the Backup Output File

All Oracle RMAN backup operations produce a single database backup file. The default file extension is .rbf, but you can specify another file extension on the RMAN Backup command line.

The following sections provide information about writing the backup file to tape or disk devices, and the contents and size of the resulting backup file.

7.3.2.1 Writing Output to Multiple Tapes

The backup operation creates one file on multiple volumes that follow the ANSI tape standards. An ANSI labeled tape is a sequential file. If all the data does not fit on one tape volume, an End-Of-Volume record is written at the end of the tape and the data continues on the next volume.

When reading the tape, the End-Of-Volume record is a signal that the file is continued on another volume. The end of the file record is written on the last volume after all data has been written on the last volume. Data from different files cannot be intermixed. All bytes for one file must appear on the tape followed by all bytes for a second file and so on. Oracle RMU does not support starting one file followed by the start of another file on the same tape.

Note

When Oracle RMU creates a multiple-volume backup file, you can append data only to the end of the last volume. You cannot append data to the end of the first or any intermediate volumes.

When multiple tape volumes are available for the backup operation, Oracle RMU does not write the tape volumes sequentially. Instead, Oracle RMU writes to the multiple volumes at the same time to gain better performance. When the backup operation is complete, it has created a multi-volume set that appears to have been created by writing sequentially to a tape. When a tape is full, but not all data is written, the backup operation writes out an End-Of-Volume label to that tape, regardless of how much tape is left on that volume. Furthermore, additional data cannot follow an End-Of-Volume record on a tape.

If your goal is to:

- Pack as much data onto the tapes, back up each database to only one of the two drives. Of course, you can perform two backup operations to two different tape drives at the same time.
- If your goal is to back up the database as quickly as possible, then you may have unused tape on all but the last volume of your backup.

7.3.2.2 Output to Tape or Disk

Although Oracle RMU can back up the database file to either disk or tape media, Oracle Corporation recommends storing your database backup file directly onto one or more tapes because:

- Tapes allow you to decrease the time it takes to perform a backup operation because you can write to several tapes in parallel. You can write to only a single disk device at a time. The time it takes to perform a backup operation to a single disk is about the same as it is to a single tape.
- Tapes are expandable to an infinite length. If you need more space, you simply mount another tape volume. However, when backing up to disk, the size of your backup file is limited to the size of the disk. Disks are finite-sized containers. (You can increase the size of disks somewhat by using RAID (redundant arrays of inexpensive disks) striped disks or bound volumes (for OpenVMS).)
- Tapes are easier to store, thus tapes make a better archival medium.
- Disks are at risk to damage from hardware failures, such as a head crash.

Note

Parallel backup operations require that you back up to tape devices.

If the database is very small, you might choose to write the backup file to a single disk file. The following table provides recommendations for tape and disk backup operations.

If You Write the Backup File to . . .	Then . . .
Disk	Store the file on a disk other than the one on which your database resides. You do not want to risk losing both your database and your backup files if you lose a disk. See Section 7.12 for more information about backing up to a disk device.
Tape	Create and store the database backup file directly onto one or more tape devices, or you can create and store it on a disk device and later back up the file to tape using the OpenVMS Backup utility or the Digital UNIX <code>tar</code> function. See Section 7.13 for more information about backing up to tape devices.

7.3.2.3 Contents and Size

Although backup files incur some overhead to support the backup format, they are typically smaller than the actual database.

Note

Do not use the size of the backup file as an indication of the size of the database files. Use the `RMU Analyze` command to determine the actual data content. See the *Oracle RMU Reference Manual* for information.

The backup file is smaller than the actual database (for most backup operations), because available space in the database root file and empty database pages in storage area files are not written to the backup file. They can be reconstructed with a restore operation.

Although the Oracle RMU backup operation does not write backup files in compressed format, you can enable hardware data compression when you use tape drives as your backup output device. (Refer to your operating system documentation for information about enabling data compression.) Data compression provides the best backup performance to tape. You cannot enable data compression when backing up your database to disk devices.

Table 7–2 summarizes which files, storage areas, and pages are backed up for each type of Oracle RMU backup operation. The database root file is backed up in every type of backup operation, including a by-area backup operation.

Table 7–2 Files, Areas, and Pages That Oracle RMU Backs Up

Type of Backup	Files, Areas, and Pages That Are Backed Up				
	.rdp Files	.rda Files	Empty Pages	SPAM Pages	.snp Files
Full and complete	Yes	All database pages in all storage areas in a database	Yes (Compressed)	No	No
Full and by-area	Yes	1	1	1	1
Incremental and complete	Yes	Only database pages in storage areas that have changed since the last full backup	Yes (if the page became empty since the last backup operation)	No	No

¹All the database pages in storage areas you specify with the Include or Exclude qualifier

(continued on next page)

Table 7–2 (Cont.) Files, Areas, and Pages That Oracle RMU Backs Up

Type of Backup	Files, Areas, and Pages That Are Backed Up				
	.rdb Files	.rda Files	Empty Pages	SPAM Pages	.snp Files
Incremental and by-area	Yes	²	²	²	²

²Only database pages in storages areas that have changed since the last full backup or you specify with the Include or Exclude qualifier

7.3.2.4 Protection

You can use the Protection qualifier on the RMU Backup command to change the default file protection for the backup file. If you do not specify the Protection qualifier, it defaults as follows:

- On OpenVMS systems, the default protection is:
 - S:RWED,O:RE,G,W for disks
 - S:RW,O:W:RW,G:RW,W:R for tapes

Tapes do not allow delete or execute access, and SYSTEM always receives read and write access.
- On Digital UNIX systems, the default protection for tapes is `-rw-r-----`

See the *Oracle RMU Reference Manual* for more information.

7.4 Why Not Use Other Backup Utilities?

Oracle Corporation recommends that you use only the Oracle RMU utility to back up Oracle Rdb databases. *Do not* use other backup utilities such as the OpenVMS Backup (BACKUP) utility or the Digital UNIX `tar` function to back up and restore Oracle Rdb databases, *especially* multfile databases.

The information in Table 7–3 describes how the Oracle RMU utility automates backup tasks that you would have to perform manually with operating system backup utilities.

Table 7–3 Comparison of Oracle RMU Backup and Operating System Backup Utilities

Category	Oracle RMU Backup	Operating System Backup
Concurrency	Controls (allows or prevents) database access from users during the backup operation.	Locks one file at a time, possibly either causing other users to fail or producing a corrupt backup copy, because a transaction is recorded for one area but not for another area.
Efficiency	Allows you to back up only the files needed to re-create the database. For example, you might exclude free spaces on used pages, snapshot records, or database structures that can be reproduced with an Oracle RMU restore operation.	Requires that you back up all files associated with the database.
Flexibility	Allows you to restore (using the RMU Restore command) the database to different locations, and change database characteristics during a restore operation. Also, the RMU Restore command allows you to restore only the pages that are corrupt, while the rest of the database is in use.	Is unable to relocate files or change database characteristics. This is because the operating system backup utilities do not contain internal pointers to the various files.
Incremental backup	Provides an incremental backup of database pages that have changed since the last full backup.	Provides incremental backup at the file level only.
Integrity	Automatically: <ul style="list-style-type: none"> • Updates the database root file with the correct location and version number of all the files associated with the database. • Recovers all the necessary recovery-unit journal files before starting the backup operation. • Works in conjunction with after-image journaling. 	Requires that you manually: <ul style="list-style-type: none"> • Back up the database root file; operating system backup utilities do not automatically update the database root file. • Check for (and recover) any outstanding recovery-unit journals before starting the backup operation. • Keep track of the locations for the after-image journal (.ajj), database root (.rdb), storage area (.rda), and snapshot (.snp) files.¹
Online /Offline	Backs up databases while they are on line and accessible by users and applications.	Requires that all database activity be stopped during the backup operation. ²

¹When you perform an Oracle RMU restore operation, you *must* know the disk device and directory of each file specification. If these files were created with concealed logical names, then you also *must* know these logical names. If you have a multifile database, this can become very complicated. One missing file following an OpenVMS restore operation will cause the database to be corrupt.

²For cluster environments, make sure there are no users on the database on *any* node in the cluster environment during the backup operation. Also, the database must be closed if global buffers are enabled.

(continued on next page)

Table 7–3 (Cont.) Comparison of Oracle RMU Backup and Operating System Backup Utilities

Category	Oracle RMU Backup	Operating System Backup
Processing	Reads simultaneously from all disk drives used for the database to avoid an I/O bottleneck on one disk drive. Writes simultaneously to multiple tape drives to keep the tapes running at “streamlining” speed.	Reads from one disk drive. Writes to one tape drive at a time.
Speed	Is designed specifically for backing up Oracle Rdb databases, and is optimized for database backup operations. Oracle RMU is significantly faster than operating system backup utilities.	The OpenVMS Backup utility processes approximately 0.5 gigabytes per hour. The Digital UNIX tar function statistics are not available.
Support	Is the only supported backup utility for Oracle Rdb databases.	Is not supported for Oracle Rdb databases.

In addition, if you use OpenVMS or Digital UNIX backup utilities, the Oracle Rdb database must not require recovery when the backup operation is completed. If it requires recovery, then the restored database is corrupt until you restore recovery-unit journal (.ruj) files. Therefore, you *must* locate *all* database recovery-unit journal files and back them up, too, as part of your backup operation because these backup utilities do not restore an unrecovered database.

Note

Oracle Corporation also recommends against using OpenVMS shadowed disks or Digital UNIX mirrored (LSM) disks as an alternative strategy to Oracle RMU database backup. These products are intended for data availability, not data integrity. Using shadowed or mirrored volumes to back up an Oracle Rdb database is prone to errors and can result in extended periods of database inaccessibility.

7.5 Database Backup Strategies

Your backup strategy should find the proper balance between these objectives:

- Minimize the operational overhead of the backup operations
- Maximize database security and integrity

To help you achieve this balance, you should take into account the requirements of your applications and users, the importance and volatility of the data, and the resources you can devote to performing the backup operation.

Also, keep in mind the amount of file management and manual intervention required with each type of backup operation. For example, there is relatively little manual intervention required for a full and complete backup operation compared with a by-area backup operation. Although you gain backup flexibility, a by-area backup operation requires that you have a sound backup file-management strategy that is coordinated with your after-image journaling procedures.

The sections referenced in the following table provide discussions to help you weigh the specific requirements of your database against the capabilities of the different types of Oracle RMU backup operations.

Section	Reference
Determining how frequently you need to back up your database	Section 7.5.1
Requirements for using full and complete backup operations	Section 7.5.2
Choosing between a full versus an incremental backup operation	Section 7.5.3
Choosing by-area backup operations	Section 7.5.4
Performing backup operations to online versus offline databases	Section 7.5.5
Using parallel backup operations	Section 7.5.6

7.5.1 Determining How Frequently You Need to Back Up Your Database

The type and frequency of your backup operations depend on the size of your database, the frequency of its update activity, and your strategy for restoring a database in the event of a failure. Table 7–4 describes some of these considerations.

Table 7–4 Determining the Frequency of Database Backups

If Your Database . . .	Then . . .
Is small	Perform full and complete backup operations only, because they take little time or disk space.
Is large	Perform a full and complete backup operation once a week and incremental backup operations every working day.
Is updated infrequently	Perform backup operations infrequently or perform frequent incremental backup operations.
Is updated daily	Back up the database daily.

(continued on next page)

Table 7–4 (Cont.) Determining the Frequency of Database Backups

If Your Database . . .	Then . . .
Contains read-only or write-once storage areas	See Section 7.9 to further modify your backup strategy to match your particular application.

7.5.2 Requirements for Using a Full and Complete Backup Operation

A full and complete database backup operation is the default Oracle RCU backup operation. This option copies the database root file and all storage areas in the database. You must perform a full and complete backup operation under the following circumstances:

- Before you perform an incremental backup operation for that database. Your first backup operation must be a full backup to create an initial database backup file. Similarly, you must restore your database from a full backup file before you can restore the latest incremental backup file for that database since the last full backup operation was made.
- After you make changes in the physical or logical design. If you were to perform an incremental backup operation under these circumstances, it could result in the inability to recover the database properly.
- Prior to installing (upgrading to) a newer version of Oracle Rdb on your system.

Note

You cannot apply a by-area backup operation, an incremental backup operation, or a set of after-image journals across an Oracle Rdb version upgrade. Therefore, you must always perform a full and complete backup operation prior to installing and upgrading the version of your Oracle Rdb software.

- After using the RCU Convert command to convert an Oracle Rdb database.
- Before and after you use the RCU Repair command.
- After using an RCU Alter command that changes the contents or file names in a database (for example, after the Commit, Deposit, Deposit Root, or Move commands).

Section 7.7 describes how to perform a full and complete backup operation.

7.5.3 Guidelines for Choosing a Full Versus an Incremental Backup Operation

Use the guidelines in Table 7–5 to determine whether to perform a full or incremental database backup:

Table 7–5 Determining When to Use a Full or an Incremental Backup Operation

Perform ...	If ...
A full backup operation	<p>10 percent of the database pages have changed since the last full backup was performed.</p> <p>Over time, you can readjust this backup strategy to include more incremental backups according to your particular database usage. For example, you might modify your strategy based on the following metrics:</p> <ul style="list-style-type: none"> – Track the elapsed time required to do an incremental backup each day. Because an incremental backup requires more calculations than does a full backup, it can actually take longer to complete. – If the time required to do an incremental backup is close to the time required to do a full backup, you should perform a full backup instead.
A full backup operation	<p>Any of the following conditions exists:</p> <ul style="list-style-type: none"> – After you finish loading data in a new database – After a large percentage of the data has changed due to some special processing – Before and after changing the structure of your database (with the SQL ALTER DATABASE statement) – Before upgrading versions of Oracle Rdb – After correcting any sort of database corruption – After using the RMU Convert command to perform a database conversion
A full backup operation	<p>The most important consideration is to minimize recovery time.</p> <p>In some cases, a large incremental database restoration might take longer to complete than a full backup restoration. Be sure to measure the time required to restore the largest incremental backup in your strategy.</p>
More incremental backup operations	<p>The most important consideration is to minimize the impact on users and applications during backup operations.</p>

Verify that your backup strategy is adequate by practicing a complete Oracle RMU restore operation including after-image journal recovery. This will help

you determine if the time required to restore the database is acceptable in your business environment.

For more information, see:

- Section 7.7 for information about starting a full backup operation
- Section 7.8 for information about starting an incremental backup operation

7.5.4 Guidelines for Choosing a By-Area Backup Operation

When you implement a physical database design that uses updatable (read/write and write-once) and read-only storage areas, you may want to implement a more flexible **by-area** backup strategy that backs up only selected storage areas.

7.5.4.1 Strategies

You can save considerable time and resources by modifying your backup strategy to perform full/by-area and incremental/by-area backup operations. This is especially true for large to very large databases and for databases that are mostly read-only or that use WORM media almost exclusively.

One of the best policies to observe when devising your backup strategy is to *plan carefully for database recovery*. Because the backup file may be a partial by-area backup operation, you must ensure that you can restore and recover all storage areas of a database. A well-planned by-area backup strategy:

- Does not omit any storage areas
- Takes into account the activity within all tables for each storage area
- Ensures the database can be completely restored
- Recovers the database up to the most recently completed transaction

A by-area backup operation allows you to choose which storage areas (read/write, read-only, and write-once) are backed up when you enter the RMU Backup command.

Table 7–6 Strategies for Backup File Management

Storage Areas	Strategy
Read-Only and Write-Once	<p data-bbox="256 709 1084 741">For read-only and write-once storage areas, consider the following backup strategy:</p> <ul data-bbox="256 758 1289 1253" style="list-style-type: none"><li data-bbox="256 758 1289 898">• Perform a full and complete backup operation on the database on a regular basis so that you have one backup file that contains the entire database. <p data-bbox="305 821 1289 898">An exception to this recommendation is when a database contains hundreds of write-once, read-many (WORM) disk media and only a few WORM drives. In this case, it is impractical to perform a full and complete backup operation on the database.</p><li data-bbox="256 915 1289 1182">• Back up selected collections of storage areas. <p data-bbox="305 951 1289 1052">A collection consists of areas associated by data interrelations, or by physical data placement (for example, data residing on the same disk, I/O controller, striped disks, and so forth.) How frequently you back up these collections depends on how frequently the collection is updated and how much time is allowable for restoration and recovery of data in the collection.</p><p data-bbox="305 1062 1289 1182">For example, suppose you have five areas on one I/O controller. If the I/O controller goes bad, you need to restore five areas. It is faster, easier, and more cost-effective to do this as one backup operation rather than restoring five areas from multiple tapes or performing five separate restore operations from five backup operations. Also, keeping related data (data that is updated as a unit) together makes sense if you ever need to reconstruct an historical state from backup files.</p><li data-bbox="256 1199 1289 1253">• Back up write-once and read-only storage areas with a frequency that depends on the limits of restore and recovery time if the entire database or write-once or read-only areas are lost.

(continued on next page)

Table 7–6 (Cont.) Strategies for Backup File Management

Storage Areas	Strategy						
Read-Only	<p>Once you set the storage areas to read-only and back them up, you no longer need to include them as part of your regular backup strategy. You need to back up only the updatable storage areas on a regular basis. Use the Include or Exclude qualifier to explicitly select read-only storage areas.</p> <p>Using this strategy, you might be able to back up fewer storage areas and possibly realize some time and resource savings in the backup operation. The savings are dependent on the size of the read-only storage areas. For example:</p>						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; width: 50%;">If . . .</th> <th style="text-align: left; width: 50%;">Then . . .</th> </tr> </thead> <tbody> <tr> <td>Your database application has read-only access exclusively or contains many large read-only storage areas</td> <td>The savings can be considerable.</td> </tr> <tr> <td>Your database application has a few small read-only storage areas, or none at all</td> <td>The savings may be minimal.</td> </tr> </tbody> </table>	If . . .	Then . . .	Your database application has read-only access exclusively or contains many large read-only storage areas	The savings can be considerable.	Your database application has a few small read-only storage areas, or none at all	The savings may be minimal.
If . . .	Then . . .						
Your database application has read-only access exclusively or contains many large read-only storage areas	The savings can be considerable.						
Your database application has a few small read-only storage areas, or none at all	The savings may be minimal.						
	<p>For best results, you must:</p> <ul style="list-style-type: none"> • Enable after-image journaling to keep track of all committed transactions between backup operations • Retain after-image journals for a sufficiently long period so you can recover the storage areas with the lowest frequency of backup <p>You must recover read-only storage areas from after-image journals because read-only access can be changed at any time. So, you must use the journals to determine if the access was changed since the last backup operation. If some area is backed up only once a month, then you must keep journals for at least 1 month. Keep journals for 2 to 3 months for the best failure protection.</p>						

(continued on next page)

Table 7–6 (Cont.) Strategies for Backup File Management

Storage Areas	Strategy										
Read/Write	<p>If you know that one or more read/write storage areas remain unchanged for extended periods of time and you do not want to make these storage areas read-only, you can implement the following alternative backup strategy:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Step</th> <th style="text-align: left;">Task</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">1</td> <td>Enable after-image journaling to keep track of all committed transactions that occur between backup operations.</td> </tr> <tr> <td style="text-align: center;">2</td> <td>Monitor table or storage area update activity.</td> </tr> <tr> <td style="text-align: center;">3</td> <td>Modify your backup schedule by selectively excluding any unchanged storage areas from the regular weekly or daily backup operation. (See Section 7.5.4.2 for selecting storage areas.)</td> </tr> <tr> <td style="text-align: center;">4</td> <td>Retain after-image journals for a sufficiently long period of time so that you can recover the storage area with the lowest frequency of backup.</td> </tr> </tbody> </table> <p>If you implement this backup strategy, plan carefully so that all read/write storage areas containing new or updated table rows are backed up regularly.</p>	Step	Task	1	Enable after-image journaling to keep track of all committed transactions that occur between backup operations.	2	Monitor table or storage area update activity.	3	Modify your backup schedule by selectively excluding any unchanged storage areas from the regular weekly or daily backup operation. (See Section 7.5.4.2 for selecting storage areas.)	4	Retain after-image journals for a sufficiently long period of time so that you can recover the storage area with the lowest frequency of backup.
Step	Task										
1	Enable after-image journaling to keep track of all committed transactions that occur between backup operations.										
2	Monitor table or storage area update activity.										
3	Modify your backup schedule by selectively excluding any unchanged storage areas from the regular weekly or daily backup operation. (See Section 7.5.4.2 for selecting storage areas.)										
4	Retain after-image journals for a sufficiently long period of time so that you can recover the storage area with the lowest frequency of backup.										

7.5.4.2 Command Qualifiers

You select storage areas using the Include, Exclude, Read_Only, and Worm qualifiers with the RMU Backup command. A by-area backup operation may be either full or incremental.

See Section 7.9 for more information about starting a by-area backup operation.

7.5.5 Guidelines for Performing Online Versus Offline Backup Operations

Oracle RMU can make a complete and consistent backup copy of your database either while the database is on line while users have access to the (open) database, or off line while access to the database is denied (closed) to users.

Table 7–7 compares the advantages of using online versus offline backup operations.

Table 7–7 Comparison of Online and Offline Backup

Perform Offline Database Backup When . . .	Perform Online Database Backup When . . .
There are periods in the day or night when there is no demand for database access and operators are available to mount the backup media. ¹	Applications or users must have uninterrupted access to the database, 24 hours a day.
You do not want to add the CPU and I/O demands of backup to the existing database work load at any time during regular processing periods.	Your system has sufficient capacity to add the backup work load to the existing user work load during your processing cycle.
The update activity on your database is heavy and you are unwilling to tolerate the snapshot file growth that results from online backup operations.	You have sufficient disk space available for the snapshot file growth that occurs during the long backup read-only transaction.
You make extensive use of Batch Update or Exclusive transactions.	
You must disable snapshot files.	You can enable snapshot files.

¹If you start an offline backup operation while users are attached to the database, the backup operation fails (see Example 7–10).

If you choose to perform a backup while the database is off line, you must restrict access to the database using the RMU Close command before you begin the backup operation. After the backup operation has completed, open the database manually or automatically when you finish the backup operation. (Chapter 4 describes using the RMU Open and Close commands.)

See Section 7.11 for information about performing an online backup operation.

7.5.6 Guidelines for Choosing Parallel Backup Operations

OpenVMS OpenVMS
VAX Alpha

Parallel backup refers to multiple processes cooperating to create one backup file. The processes run in parallel to reduce the time required to back up a database. The master process, called a **coordinator process**, controls **worker processes** that perform the backup operation. The worker processes concurrently perform backup operations using the same algorithm used for the default (non-parallel) Oracle RMU multithreaded backup. On cluster systems, the worker processes can run on different nodes in the cluster.

In most cases, the default (non-parallel) Oracle RMU backup operation is capable of sufficiently high performance to satisfy most database backup requirements. The default Oracle RMU backup operation can drive all I/O paths to capacity up to the limit of the CPU.

However, if your backup performance is not satisfactory, the first step is to check your CPU utilization as described in the following table:

If . . .	Then . . .
There is less than 90% CPU utilization during the backup operation	There is a bottleneck in the I/O subsystem, or you do not have enough tapes running at once, or storage areas are not spread over enough disks. Check individual and aggregate bandwidth capacities for all buses, adaptors, controllers, and devices that are used during the backup operation. Use the OpenVMS Monitor utility (MONITOR /SYSTEM) or the Digital UNIX <code>vmstat</code> command to show CPU utilization. Verify that the bandwidth capacities are sufficient to support the expected backup performance.
The problem is not due to the I/O subsystem and the CPU utilization is high	It is likely that the demands of the backup operation exceed the capacity of a single CPU. Although powerful, the default (non-parallel) Oracle RMU backup operation executes as a single process and can exceed the capabilities of a single CPU. Use the Oracle RMU parallel backup operation to overcome this performance barrier.

The parallel backup operation can be useful in the following situations:

When . . .	Then . . .
You back up the database to multiple tape drives and the tape drives cannot run at full speed because there is not enough CPU power	You can use parallel backup to distribute the work across multiple CPUs. For example, if you have multiple CPUs (either because you are using an SMP machine or you are running in a cluster environment) and there is unused time on other CPUs.
You need to perform backup operations as quickly as possible	You can add more tape drives and CPUs to do the work in parallel processes.
Your system configuration includes multiple CPUs	Parallel backup allows performance to scale across CPUs in an SMP system and across nodes in a cluster. Because each worker process is a powerful multithreaded backup thread, there is no need to configure more than one worker process per available CPU. By configuring fewer worker processes than CPUs, you can ensure that CPU capacity remains for other work besides the backup operation.
You want to partition work among specific worker processes	Parallel backup enables you to assign the processes to nodes that can most swiftly accomplish the work.

When . . .	Then . . .
You have multiple nodes in a cluster and you have tape drives that are local to the nodes	You can use parallel backup to assign worker processes so that they write to local devices. (For example, if tape drives are connected locally to each node in a cluster.) This backup strategy performs better than a configuration in which worker processes write to devices that are served remotely. The backup operation can write to local tapes faster than served tapes, so it is faster to assign each worker process to the same node as the tape drive to which the worker process writes.
You want to monitor the performance of your backup operation on a Windows system, you can run with only one parallel process	If you specify only one worker process on the RMU Backup command, Oracle RMU performs a default (non-parallel) backup operation. In this case, the backup operation still invokes a coordinator and a worker process. By specifying a non-parallel backup operation in this way, you can monitor the progress of your non-parallel backup operation on your Windows system using the Parallel Backup Monitor Windows interface.

The parallel backup process requires that you create a plan file that directs the work of the backup processes.

See Section 7.10 for more information about creating a plan file and starting a parallel backup operation. ♦

7.6 Implementing a Reliable Database Backup Procedure

This section provides general recommendations for achieving database integrity and includes a sample procedure for creating and maintaining regular database backup operations.

In addition, Section 7.6.3 describes how to calculate minimum working set requirements, and Section 7.6.4 explains how to use the `Checksum_Verification` qualifier to verify each database page before it is backed up.

7.6.1 Recommendations

Using the guidelines in Table 7–8, you should be able to develop an overall strategy to help you recover from the failure of any single component of the database.

Table 7–8 Recommendations for Safeguarding Database Integrity

Prior to Backup	During Backup	After Backup
Consider:	Routinely:	Periodically:
<ul style="list-style-type: none"> Spreading the database files over many disk drives. Using after-image journaling to retain completed transactions. Making sure the after-image journal file is on a separate disk from the rest of the database. Implementing RAID (redundant arrays of inexpensive disks) technology (such as disk mirroring or shadowing) to prevent a disk device failure from interrupting system and application operations. 	<ul style="list-style-type: none"> Verify that the database is internally consistent and can be restored if necessary. Back up the after-image journal before you back up the database. Back up the database at regular intervals. Schedule backup operations during periods with the lowest database activity. Maintain a log of backup activities. This log should contain elapsed times for each backup activity. Use backup journal files to record details of your backup operations. Check the backup file and media using the RMU Backup and RMU Dump commands. If possible, separate OpenVMS and Digital UNIX backup operations from Oracle RMU backup operations to avoid contention for disk and tape drives, and to avoid mixing up or mislabeling tape volumes. Do not use the OpenVMS Backup utility or the Digital UNIX <code>tar</code> function to back up your Oracle Rdb database. 	<ul style="list-style-type: none"> Restore and recover the database to a new set of disks and verify the database copy. Store some backup volumes to a location separate from where you store the database root file. Test your backup strategy by simulating a failure.

7.6.2 Sample Backup Procedure

Table 7–9 takes you step-by-step through a sample backup procedure. The sample demonstrates a procedure that is useful for a database that does not contain read-only or write-once storage areas. (See Section 7.9 for information about working with databases that contain read-only or write-once storage areas.)

Assume the following for the sample procedure described in Table 7–9:

- The backup operation is performed on a large database that is updated daily.
- You use the RMU Backup command to perform a full and complete database backup operation each Friday.

You might also consider using a verify operation with the full backup operation. Whether you perform a verify operation depends on the size of the database, the time available to verify the database, and your system configuration and resources. For larger databases, the verify operation is not practical. In some cases, the verify operation can take longer to perform than the backup operation itself.

- You use the RMU Backup Incremental command to perform incremental database backup operations on the other working days, Monday through Thursday.
- If anything happens to your database, you can restore a full or an incremental database backup file from tape. Use the RMU Restore, and the RMU Restore Incremental commands to restore your database to the condition it was in at the time of the most recent database backup operation.

Table 7–9 Sample Full and Complete, and Incremental Backup Procedures

Step	Action
❶	Backup after-image journals. Refer to Chapter 9 for more information.
❷	Start a full backup operation. The following examples demonstrate the commands for OpenVMS and Digital UNIX: <pre>\$ RMU/BACKUP DB_DISK:[MFPERS]MF_PERSONNEL DBS_BACKUPS:PERS_FULL.RBF \$ rmu -backup /usr/db_disk/database/mf_personnel.rdb /usr/pers_full.rbf -log</pre>
❸	Verify the contents of a backup file after a backup operation. Optionally, you can check the readability of the backup (.rbf) file by using the RMU Dump command with the Backup_File qualifier. This step is recommended if you experience problems with hardware or media during the backup operation and want to check the internal structure of the backup file for readability. If you do not specify the Options qualifier, Oracle RMU returns the operating system prompt, indicating that the backup file (.rbf) file is fine; otherwise, Oracle RMU displays an error message. The following examples demonstrate the commands for OpenVMS and Digital UNIX: <pre>\$ RMU/DUMP/BACKUP_FILE DBS_BACKUPS:[MFPERS]PERS_FULL.RBF \$ rmu -dump -backup_file /usr/pers_full.rbf</pre>

(continued on next page)

Table 7–9 (Cont.) Sample Full and Complete, and Incremental Backup Procedures

Step	Action
4	<p>Verify a database prior to an incremental backup operation.</p> <p>Each regular weekday night (Monday through Thursday), you might want to perform an incremental database verify operation (to ensure that the database is not corrupt) prior to performing an incremental backup operation.</p> <p>The following examples demonstrate the commands for OpenVMS and Digital UNIX:</p> <pre>\$ RMU/VERIFY/INCREMENTAL DB_DISK:[MFPERS]MF_PERSONNEL \$ rmu -verify -incremental /usr/db_disk/database/mf_personnel.rdb</pre>
5	<p>Start an incremental backup operation if the incremental verify operation is successful (no error messages are displayed that indicate that verification has failed).</p> <p>The following examples demonstrate the commands for OpenVMS and Digital UNIX:</p> <pre>\$ RMU/BACKUP/INCREMENTAL DB_DISK:[MFPERS]MF_PERSONNEL DBS_BACKUPS:[MFPERS]PERS_INCR.RBF \$ rmu -backup -incremental /usr/db_disk/database/mf_personnel.rdb /usr/pers_incr.rbf</pre>
6	<p>On OpenVMS systems, limit the number of incremental backup file versions on your system by:</p> <ul style="list-style-type: none">Setting purge limits for incremental backup files. For example: <pre>\$ SET FILE/VERSION_LIMIT=2 DBS_BACKUPS:[MFPERS]PERS_INCR.RBF</pre><p>You can purge your incremental backup file after each incremental backup operation or use the DCL SET FILE command, as shown, to automatically maintain a specific number of versions of your incremental backup file. Keep the latest two versions of the backup file in case one is damaged.</p>Using different file names (for example, use PERS_INCR_MONDAY.RBF on Monday).
7	<p>Verify the contents of an incremental backup file.</p> <p>Verifying the backup file is not a requirement; it is necessary only if you experience problems during the backup operation.</p> <p>The following examples demonstrate the commands for OpenVMS and Digital UNIX:</p> <pre>\$ RMU/DUMP/BACKUP_FILE DBS_BACKUPS:[MFPERS]MF_PERSONNEL PERS_INCR.RBF \$ rmu -dump -backup_file /usr/db_disk/database/mf_personnel.rdb /usr/pers_incr.rbf</pre> <p>If the operating system prompt displays and there are no error messages, the internal structure of the incremental backup file is intact.</p>

OpenVMS OpenVMS See Section 7.13 for information about allocating and mounting tape drives on
VAX Alpha OpenVMS systems during the backup procedure. ♦

7.6.3 Computing Working Set Requirements for OpenVMS

OpenVMS OpenVMS
VAX Alpha

You can optimize database backup performance on OpenVMS systems by determining the minimum working set requirements and setting the OpenVMS process quotas accordingly.

How to Calculate the Minimum Working Set

Table 7–10 describes how to determine working set sizes and includes examples of calculations that are based on qualifier values from the following RMU Backup command:

```
$ RMU/BACKUP/LOG/REWIND/NOCRC/ACTIVE_IO=5/BLOCK_SIZE=65024 -
_$/PAGE_BUFFERS=2/JOURNAL=MONDAY.JNL/LABEL=(TAPE01) -
_$/MF_PERSONNEL.RDB T1:MONDAY_MF_PERS.RBF/MASTER
```

The following list uses the values of the Page_Buffers, Active_IO, and Block_Size qualifiers to demonstrate the working set calculations for a database with 50 storage areas:

1. Calculate minimum working set requirements using the equations shown in Table 7–10 (note that the units are in 512-byte pagelets):

Table 7–10 Calculating Working Set Size

Step	Calculate:	Example:
❶	Start with the value 1,000.	1000
❷	Multiply the number of storage areas being backed up * 164 (blocks). (For a by-area backup operation, this is less than the total number of storage areas in the database.)	50 * 164 = 8200
❸	If Page_Buffers ¹ is equal to 3, 4, or 5, multiply the number of storage areas being backed up * (Page_Buffers - 2) * 64 (blocks).	(Because Page_Buffers=2 in the example, you can skip this step.)
❹	Multiply the number of master tape drives * Active_IO * (Block_Size / 512). ^{2, 3}	1 * 5 * (65024 / 512) = 635
❺	Add all the values in the “Example:” column to compute the appropriate working set value for the sample backup procedure.	1000 + 8200 + 635 = 9835
❻	On OpenVMS Alpha systems, round the working set size up to the nearest multiple of 16 (there are 16 blocks in an 8K OpenVMS Alpha page).	(Round 9835 to the nearest multiple of 16) = 9840

¹The Page_Buffers qualifier specifies the number of disk buffers (from 2 to 5) assigned to each storage area.

²Active_IO specifies the maximum number (from 1 to 5) of simultaneous write operations to a backup device.

³Block_Size specifies the maximum record size (usually 32K or 64K) for the backup file.

2. Use the working set size calculated in Table 7–10 and the equations shown in Table 7–11 to calculate the OpenVMS process quota values for the account where you enter the RMU Backup command:

Table 7–11 Recommended OpenVMS Process Quotas

OpenVMS Process Quota	Calculate:	Example:
WSDEF	Working set size	9840 ¹
WSQUOTA	Working set size + 2048	11,888
WSEXTENT	Working set size + 4096	13,936

¹Calculated in Table 7–10

Using these recommended settings ensures adequate memory for the backup process and minimizes paging.

3. Use the OpenVMS Authorize (AUTHORIZE) utility to display the current quota values for the account you use for backup operations. Change the values, if necessary.
4. Log out of the account and then log in again so that the process quotas can take effect.

Note

Parallel backup operations might require larger working set quotas than non-parallel backup operations. For example, a worker process in a parallel backup operation might require a larger working set than a non-parallel backup operation that backs up the same number of storage areas. The calculations described here provide a good starting point for estimating the working set requirements of [RMU\$SRV70] (the account under which a worker process runs during a parallel backup operation). If you observe excessive page faulting, you can increase the quotas.

Refer to the *Oracle RMU Reference Manual* for additional information about the Page_Buffers, Active_IO, and Block_Size qualifiers. ♦

7.6.4 Checking the Database Page Checksum During a Backup to Disk

The RMU Backup command provides checksum verification to authenticate each database page before it is backed up. Using the Checksum_Verification qualifier permits Oracle RMU to detect read failures and correct them, resulting in:

- A comprehensive error detection capability during the read operation of the backup procedure

- Added reliability when you are experiencing disk hardware problems

Note

The Checksum_Verification qualifier verifies database pages only if you enabled checksum calculations with any of the following SQL statements:

- ALTER DATABASE CHECKSUM CALCULATION IS ENABLED
 - CREATE DATABASE CHECKSUM CALCULATION IS ENABLED
 - CREATE STORAGE AREA CHECKSUM CALCULATION IS ENABLED
-

Although checksum verification uses significant CPU resources, it can provide an extra measure of confidence in the quality of data that you back up. Use the guidelines in the following table to determine when to enable and disable the checksum capability:

Qualifier	Description
Nochecksum_Verification	Use for offline backup operations. This is the default setting. For offline operations, the additional CPU cost of using the Checksum_Verification qualifier might not be justified unless you are experiencing disk, controller, or port hardware problems.
Checksum_Verification	Use for all online operations that include database backup operations, moving an area, or copying a database. Such operations, when performed on line, may read a partial database write operation. This is particularly true with shadow sets and striped disks. These technologies distribute data over several disk drives, and use of the Checksum_Verification qualifier permits Oracle RMU to detect the possibility that the data it is reading from these disks has only been partially updated.

Example Checksum Operations

When Oracle Rdb detects a disk read error or a bad page checksum, it marks the database page as corrupt. If one or more pages in a storage area have been marked as corrupt and the entries are made in the corrupt page table to denote these corrupt pages, Oracle RMU returns the error message shown in Example 7-1.

Example 7–1 Using Checksum to Verify Database Pages During a Backup Operation

```
$ RMU/BACKUP /CHECKSUM_VERIFICATION DB_DISK:[MFPERS]MF_PERSONNEL -  
_ $ DBS_BACKUPS:PERS_FULL.RBF  
%RMU-E-CORPAGPRES, Corrupt or inconsistent pages in area JOBS.RDA;1  
%RMU-F-FATALERR, fatal error on BACKUP
```

In Example 7–1, Oracle RMU cannot back up the JOBS area until the corruption is removed. In this case, the Checksum_Verification qualifier prevents you from backing up corrupt data or from having the backup operation fail later because of the corruption.

Chapter 8 describes how you can correct page corruptions using the RMU Restore and RMU Recover commands.

7.7 Performing Full and Complete Database Backup Operations

A full and complete backup operation copies the database root (.rdb) file and all the database pages in all the storage areas in the database, including empty pages. This is the default Oracle RMU backup operation. Note the following about full database backup operations:

- Oracle RMU copies enough information (into a single file) to completely reproduce all the database files and all database pages.
- You must do a full and complete backup (to create a backup file) before you can perform an incremental backup operation. Oracle Rdb reports an error if you attempt to perform an incremental backup on a database that has never been fully backed up.

When a full backup operation creates a backup file for one or more storage areas, the date and time of the last full backup file created for those storage areas is updated in the backup file. (See Section 7.14 for information about displaying backup file information.)

- When you need to recover a database from backup files, you must restore the full backup file before restoring the last incremental backup file. Each incremental backup file is associated with a specific full backup file.

In Example 7–2, a full database backup operation is performed on the mf_personnel database, copying the database root (.rdb) file and all storage area (.rda) files into a backup file named mf_pers_full.rbf. (You can specify a file type other than .rbf if you want.)

- The database root file resides on DB_DISK:[MFPERS] and the storage area files are distributed on other shared disks in this cluster.

- The database backup file resides in the directory \$SDUA9:[PERS.BACKUPS]; the logical name DBS_BACKUPS:[MFPERS] was defined for that directory. Because only a single device is specified by the logical name DBS_BACKUPS:[MFPERS], only a single thread is indicated in the log file. The database backup file also includes information necessary to create the database again and initialize snapshot files when the database is restored.

The Backup command line in Example 7-2 includes the Log qualifier to display the results of the full backup operation.

Example 7-2 Starting a Full Database Backup Operation

```
$ RMU/BACKUP/LOG DB_DISK:[MFPERS]MF_PERSONNEL.RDB -
_ $ DBS_BACKUPS:[MFPERS]MF_PERS_FULL.RBF

%RMU-I-BCKTXT_01, Thread 1 file uses devices DBS_BACKUPS:
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK1:[MFPERS]MF_PERS_DEFAULT.RDA;1
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK5:[MFPERS]SALARY_HISTORY.RDA;1
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK2:[MFPERS]RESUME_LISTS.RDA;1
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK6:[MFPERS]EMPIDS_OVER.RDA;1
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK3:[MFPERS]EMPIDS_LOW.RDA;1
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK5:[MFPERS]EMPIDS_MID.RDA;1
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK4:[MFPERS]EMP_INFO.RDA;1
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK6:[MFPERS]RESUMES.RDA;1
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK3:[MFPERS]JOBS.RDA;1
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK3:[MFPERS]DEPARTMENTS.RDA;1
%RMU-I-BCKTXT_00, backed up root file DB_DISK:[MFPERS]MF_PERSONNEL.RDB;1
%RMU-I-BCKTXT_02, full backup of storage area DISK1:MF_PERS_DEFAULT.RDA;1
%RMU-I-BCKTXT_02, full backup of storage area DISK3:[MFPERS]EMPIDS_LOW.RDA;1
%RMU-I-BCKTXT_02, full backup of storage area DISK5:[MFPERS]EMPIDS_MID.RDA;1
%RMU-I-BCKTXT_02, full backup of storage area DISK6:[MFPERS]EMPIDS_OVER.RDA;1
%RMU-I-BCKTXT_02, full backup of storage area DISK3:[MFPERS]DEPARTMENTS.RDA;1
%RMU-I-BCKTXT_02, full backup of storage area
DISK5:[MFPERS]SALARY_HISTORY.RDA;1
%RMU-I-BCKTXT_02, full backup of storage area DISK3:[MFPERS]JOBS.RDA;1
%RMU-I-BCKTXT_02, full backup of storage area DISK4:[MFPERS]EMP_INFO.RDA;1
%RMU-I-BCKTXT_02, Full backup of storage area DISK2:[MFPERS]RESUME_LISTS.RDA;1
%RMU-I-BCKTXT_02, Full backup of storage area DISK6:[MFPERS]RESUMES.RDA;1
%RMU-I-BCKTXT_02, Full backup of storage area
DISK1:[MFPERS]MF_PERS_DEFAULT.RDA;1
%RMU-I-BCKTXT_04, ignored 1 space management page
%RMU-I-BCKTXT_05, backed up 6 inventory pages
%RMU-I-BCKTXT_06, backed up 156 logical area bitmap pages
%RMU-I-BCKTXT_07, backed up 699 data pages
%RMU-I-BCKTXT_11, backup data compression ratio: 0.58
```

(continued on next page)

Example 7–2 (Cont.) Starting a Full Database Backup Operation

```
%RMU-I-BCKTXT_02, Full backup of storage area DISK5:[MFPERS]SALARY_HISTORY.RDA;1
%RMU-I-BCKTXT_04,   ignored 1 space management page
%RMU-I-BCKTXT_05,   backed up 0 inventory pages
%RMU-I-BCKTXT_06,   backed up 0 logical area bitmap pages
%RMU-I-BCKTXT_07,   backed up 126 data pages
%RMU-I-BCKTXT_11,   backup data compression ratio: 0.49
.
.
.
```

Although this example backs up the database to a disk device, Oracle Corporation recommends that you back up Oracle Rdb databases to tape devices (see Section 7.3.2.2). You can issue the same RMU Backup command to back up your database to a tape device if you include the tape device name as part of your backup file specification. For example, you would enter this command on a Digital UNIX system:

```
$ rmu -backup -log -rewind -label=back01 \  
> /usr/db_disk/database/mf_personnel.rdb /usr/mf_pers_full_bu.rbf
```

See Examples 7–11 and 7–12 for descriptions and examples that show how to write the backup file to one or more tapes on one or more tape drives.

Section 7.13.4 describes how the Oracle RMU backup operation performs its own tape label processing.

7.8 Performing an Incremental Database Backup Operation

An incremental backup operation copies the database root (.rdb) file and all database pages that have been updated since the last full backup operation into a single backup file. Free space on those pages and snapshot pages are not copied.

Note

You must perform a full backup operation before you can perform an incremental backup operation.

You can incrementally back up all database pages or only selected database pages, based on the intended use of the backup file. These backups are specified by the qualifiers shown in the following table:

Command	Description
Incremental Incremental=Complete	Produces an incremental backup file that contains only those pages updated since the last full and complete backup operation. Entering the Incremental qualifier is the same as entering the Incremental=Complete qualifier.
Incremental=By_Area	Backs up pages for each area that you have updated since the last full (full and complete, or full and by area) backup operation of that storage area. If multiple storage areas are fully backed up at different times using this option, then each storage area could have pages from a different time period. Using this backup strategy to restore multiple storage areas minimizes the volume of incremental data restored but requires the additional complexity of restoring each area from the correct full backup operation. This strategy assumes that your .rdp file is not corrupt. Bringing the database back to its most current state using by-area backup operations requires an uncorrupted .rdp file.

The incremental option always generates a backup file even when user data has not changed since the last full backup operation. Every time you perform an incremental backup operation, the new incremental backup file:

- Replaces any other incremental backup file made since the last full backup (except when you specify the Incremental=By_Area command qualifier, because backup files for specific storage areas might contain different information)
- Contains copies of all database pages changed since the last full backup operation, including pages contained in any other incremental backup files made previously

Therefore, you can restore your database to the state it was in at the time of your most recent incremental backup by first restoring the last full backup operation and then restoring the last incremental backup operation.

7.8.1 Starting an Incremental Backup Operation

Example 7-3 shows a simple incremental Backup command line for a disk backup on a Digital UNIX system. In the example, Oracle RMU returns an error message and exits the backup procedure because a full backup operation was not performed.

Example 7-3 Starting an Incremental Backup Operation

```
$ rmu -backup -incremental /usr/db_disk/database/mf_personnel.rdb \  
> /usr/mf_pers_full_bu.rbf  
%RMU-F-NOFULLBCK, no full backup of this database exists
```

Example 7-4 shows an incremental backup command on OpenVMS that creates an incremental backup file on a tape device. The tape device name is included as part of the incremental backup file specification.

Example 7-4 Mounting a Tape Volume and Starting an Incremental Backup Operation

```
$ ALLOCATE $111$MUA0:  
$ MOUNT/FOREIGN $111$MUA0: %MOUNT-I-MOUNTED, mounted on _$111$MUA0:  
$ RMU/BACKUP/INCREMENTAL /LOG /REWIND /DENSITY=6250 /LABEL=BACK01 -  
_ $ DB_DISK:[MFPERS]MF_PERSONNEL.RDB $111$MUA0:MF_PERS_INCR.RBF  
%RMU-I-BCKTXT_01, Thread 1 file uses devices $111$MUA0:  
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK1:[MFPERS]MF_PERS_DEFAULT.RDA;1  
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK5:[MFPERS]SALARY_HISTORY.RDA;1  
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK2:[MFPERS]RESUME_LISTS.RDA;1  
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK6:[MFPERS]EMPIDS_OVER.RDA;1  
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK3:[MFPERS]EMPIDS_LOW.RDA;1  
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK5:[MFPERS]EMPIDS_MID.RDA;1  
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK4:[MFPERS]EMP_INFO.RDA;1  
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK6:[MFPERS]RESUMES.RDA;1  
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK3:[MFPERS]JOBS.RDA;1  
%RMU-I-BCKTXT_08, Thread 1 was assigned file DISK3:[MFPERS]DEPARTMENTS.RDA;1  
%RMU-I-BCKTXT_00, Backed up root file DB_DISK:[MFPERS]MF_PERSONNEL.RDB;1  
%RMU-I-BCKTXT_03, Incremental backup of storage area  
DISK1:[MFPERS]MF_PERS_DEFAULT.RDA;1  
%RMU-I-BCKTXT_03, Incremental backup of storage area DISK3:[MFPERS]EMPIDS_LOW.RDA;1  
%RMU-I-BCKTXT_03, Incremental backup of storage area DISK5:[MFPERS]EMPIDS_MID.RDA;1  
%RMU-I-BCKTXT_03, Incremental backup of storage area DISK6:[MFPERS]EMPIDS_OVER.RDA;1  
%RMU-I-BCKTXT_03, Incremental backup of storage area DISK3:[MFPERS]DEPARTMENTS.RDA;1  
%RMU-I-BCKTXT_03, Incremental backup of storage area DISK5:[MFPERS]SALARY_HISTORY.RDA;1  
%RMU-I-BCKTXT_03, Incremental backup of storage area DISK3:[MFPERS]JOBS.RDA;1  
%RMU-I-BCKTXT_03, Incremental backup of storage area DISK4:[MFPERS]EMP_INFO.RDA;1  
%RMU-I-BCKTXT_03, Incremental backup of storage area DISK2:[MFPERS]RESUME_LISTS.RDA;1  
%RMU-I-BCKTXT_03, Incremental backup of storage area DISK6:[MFPERS]RESUMES.RDA;1  
%RMU-I-BCKTXT_03, Incremental backup of storage area DISK1:[MFPERS]MF_PERS_DEFAULT.RDA;1  
%RMU-I-BCKTXT_04, ignored 1 space management page  
%RMU-I-BCKTXT_05, backed up 0 inventory pages  
%RMU-I-BCKTXT_06, backed up 0 logical area bitmap pages  
%RMU-I-BCKTXT_07, backed up 0 data pages  
%RMU-I-BCKTXT_09, Est. cost to backup relative to a full backup is 0.07  
%RMU-I-BCKTXT_10, Est. cost to restore relative to a full restore is 0.01  
%RMU-I-BCKTXT_03, Incremental backup of storage area DISK5:[MFPERS]SALARY_HISTORY.RDA;1  
%RMU-I-BCKTXT_04, ignored 1 space management page  
%RMU-I-BCKTXT_05, backed up 0 inventory pages
```

(continued on next page)

Example 7–4 (Cont.) Mounting a Tape Volume and Starting an Incremental Backup Operation

```
%RMU-I-BCKTXT_06, backed up 0 logical area bitmap pages
%RMU-I-BCKTXT_07, backed up 0 data pages
%RMU-I-BCKTXT_09, Est. cost to backup relative to a full backup is 0.40
%RMU-I-BCKTXT_10, Est. cost to restore relative to a full restore is 0.01
%RMU-I-BCKTXT_03, Incremental backup of storage area DISK2:[MFPERS]RESUME_LISTS.RDA;1
.
.
.
$ DISMOUNT $111$MUA0:
$ DEALLOCATE $111$MUA0:
```

7.8.2 Optimizing Incremental Backup Performance

By default, Oracle RMU optimizes incremental backup operations by scanning regions of the database that have been updated since the last full backup operation. This optimization feature is sometimes referred to as a fast-incremental backup. The identity of changed regions is stored in the database so that, during an incremental backup operation, only these regions are scanned for updates. This optimization provides a substantial performance improvement when database activity is sufficiently low.

However, there is a cost in recording this information in the database. In some circumstances the cost might be too high, particularly if you do not intend to use incremental backup operations.

To get around these overhead costs, you can use the [No]Scan_Optimization qualifier to specify whether or not Oracle RMU should employ scan optimizations during incremental backup operations. The [No]Scan_Optimization qualifier allows you to control incremental backup optimization as follows:

- During an offline, full backup operation:

If You Specify the . . .	Then . . .
Scan_Optimization qualifier	Oracle RMU enables recording of the identities of areas that change after this backup operation completes. This is the default setting.
Noscan_Optimization qualifier	Oracle RMU disables recording of the identities of areas that change after this backup operation completes.

If you do not specify the [No]Scan_Optimization qualifier when you enter the Backup command, Oracle RMU:

1. Uses the value that you used on the previous Backup command. (That is, Oracle RMU uses the recording state set by a prior execution of the Backup command). For example, if you specified the Noscan_Optimization qualifier on the prior Backup command, the recording state remains unchanged.
 2. Uses the Scan_Optimization qualifier (by default) if you did not previously specify the Noscan_Optimization qualifier.
- During an online, full backup operation, the [No]Scan_Optimization qualifier is ignored. Oracle RMU returns an information message indicating that the [No]Scan_Optimization qualifier is ignored when you also include the Online qualifier for a full backup operation.
 - During an incremental backup operation:

If You Specify the . . .	Then . . .
Scan_Optimization qualifier	Oracle RMU uses the optimization only when Oracle Rdb has been recording the regions updated since the last full backup operation.
Noscan_Optimization qualifier	Oracle RMU does not use the optimization, regardless of whether Oracle Rdb is recording the identity of the regions updated since the last full backup operation.

7.8.3 Determining Which Pages Have Changed Since the Last Backup

The Oracle RMU backup operation automatically determines which pages have changed since the last full backup by comparing the transaction sequence number (TSN) on each storage area's SPAM pages against the TSN for the last recorded full backup operation maintained in the database root file. The incremental backup operation proceeds depending on the TSN comparison, as follows:

- When you optimize the backup operation using the Scan_Optimization qualifier:

If the TSN on the SPAM Page . . .	Then . . .
Is higher than the TSN for the last recorded full backup operation	Oracle RMU reads pages in the SPAM range for pages that have changed since the last full backup.

If the TSN on the SPAM Page . . .	Then . . .
Is lower than the TSN for the last recorded full backup operation	Oracle RMU reads every page looking for pages that have changed.

- When you do not optimize the backup operation using the `Noscan_Optimization` qualifier:

If the TSN on the SPAM page . . .	Then . . .
Is higher than the TSN for the last recorded full backup operation	A copy of those pages is written to the incremental backup file.
Is lower than the TSN for the last recorded full backup operation	The database pages in the SPAM interval are ignored.

This mechanism for comparing TSNs provides for fast, incremental backup performance compared to checking each page in the entire database to determine if the highest TSN on the page is a number larger than the TSN of the last full backup operation.

Note

An incremental backup operation on a storage area does not update the date and time for the last *full* backup operation performed on the storage area that is recorded in the backup file.

7.8.3.1 Looking at Timestamps

You can examine the TSNs manually using the `RMU Dump Header` command. For example, to determine when the last full backup operation was performed, enter the command as shown in Example 7-5.

Example 7–5 Determining the Date and Time of the Last Full Backup File

```
$ RMU/DUMP/HEADER MF_PERSONNEL
.
.
.
Latest full backup file is dated 5-NOV-1995 15:29:37.23❶
Latest full backup transaction sequence number is 76❷
Database has never been incrementally restored
Database has never been fully restored
Latest full verify occurred at 5-NOV-1995 14:07:27.72
Database has never been altered
```

The information in Example 7–5 appears at the end of the header output. Notice the following callouts:

- ❶ The last full backup operation was on November 5, 1995 15:29:37.23
- ❷ The last full backup TSN is 76

Now, enter the RMU Dump command with the Area qualifier to examine the timestamp, as shown in the command in Example 7–6.

Example 7–6 Displaying an Area to Determine a Timestamp

```
$ RMU/DUMP/AREA=EMP_INFO /START=1/END=1 DB_DISK:[MFPERS]MF_PERSONNEL
.
.
.
* Dump of storage area EMP_INFO
.
.
.
0008 00000001 0000 page 1, physical area 8 (space mgmt)
03BA800A 0006 checksum = 03BA800A
80000000 00000007 000A Fast incremental backup TSN = 7
0000 03B4 0012 948 free bytes, 0 locked
```

Pages in the SPAM interval for the SPAM page (on page 1) for the EMP_INFO storage area file are not backed up by an RMU Backup Incremental command because there have been no changes made. This is indicated by comparing the values of the TSNs in Example 7–5 (TSN value is 76) and in Example 7–6 (TSN value is 7). Because the TSN value of 7 from the timestamp is lower than the TSN value of 76 from the last recorded full backup operations, Oracle RMU does not make a backup copy of those SPAM pages.

In Example 7–6, note that you could include the Spam qualifier to dump only SPAM pages.

7.8.3.2 Checking By-Area Backup Timestamps

Timestamp information helps you determine the last time a specific storage area type (read-only, write-once, or read/write) was backed up so you can restore only one of the storage area types.

Each time the database is completely backed up, either fully or incrementally, the timestamp is updated in both the database parameters section of the backup file and the specific root record portion of the database root (.rdb) file. The timestamp is also updated for each storage area contained in the database to reflect the time of the most recent backup operation. Similarly, when a database is restored, fully or incrementally, the timestamp in the .rdb file is updated.

However, in a by-area backup operation when not all storage areas are backed up, only the timestamps for those storage areas that are backed up are updated in the backup and database root file. If one or more storage areas are restored, the timestamps for these storage areas in the .rdb file are updated. By checking these timestamps, you can determine when the database was last backed up or restored. You can also determine when each storage area was last backed up or restored.

7.8.4 Measuring the Benefits of an Incremental Backup

Output written to the backup log file provides summary statistics (estimated costs) on a per area basis on the benefit of an incremental backup or incremental restore operation compared to a full backup or restore operation.

The costs take into account the number of disk I/O operations needed and the requirement to perform the I/O operation for each area. These disk I/O costs are approximate because they do not translate directly into “clock time.” The costs do not consider:

- That each disk type has its own relative costs such as transfer rate, latency, seek time, and so forth.
- Competition for the disk by other processes.

However, these estimates can help you determine the point at which the incremental operation is becoming less productive. Use the following table to analyze the relative costs of the backup operation.

When . . .	Then . . .
The relative backup cost is greater than one	It is more efficient to perform a full backup operation than an incremental backup operation for the storage area. A value greater than one indicates that more disk I/O operations are needed to do an incremental backup operation than are needed to do a full backup operation of the area.
The backup cost approaches or exceeds one	Consider performing a full backup operation for your next area backup operation. In general, when the relative cost for an incremental by-area restore operation approaches or exceeds one, you should consider performing a full backup operation. If you consider that an incremental restore operation must always follow a full restore operation, the actual cost of restoring the area is one higher than reported. In addition, if a full restore operation takes 2 hours and you can only permit the full and incremental restore operation to take 3 hours, as the restore cost approaches 0.5 you should perform a full backup operation.

In Example 7–4, because the `mf_personnel` database is such a tiny sample database and there are no pages to incrementally back up, the estimated costs reflect some characteristics of the estimation process. The example takes a single I/O operation to determine the estimated cost for each area. However, no areas require incremental backup operations; thus the estimates reflect only the single I/O for cost estimation. This artifact should not be a problem with a large database and one that contains areas with pages updated since the last full backup operation.

7.9 Performing a By-Area Backup Operation

By default, the Oracle RMU backup operation backs up all storage areas in the database. The Oracle Rdb documentation uses the term *by-area* to refer to a backup operation that backs up the database root file and only specific storage areas in a database.

Because most hardware failures result in the loss of only a single disk drive, restoring only the storage areas on the lost drive and applying transactions since the last by-area backup presents an attractive alternative to a full database restore operation. Backing up individual storage areas greatly facilitates this kind of recovery. Section 7.5.4 discusses strategies to help you determine if this option is right for your database environment.

Caution

Do not rely on by-area backup (.rbf) files to restore an entire Oracle Rdb database. Because a by-area backup operation does not back up all storage areas, your database cannot be fully recovered. You must

perform a full backup operation (shown in Example 7–2) in order to avoid loss of all storage areas in your database.

You specify which storage areas to back up using the Backup command qualifiers shown in Table 7–12.

Table 7–12 Qualifiers for By-Area Backup Operations

Command Qualifier	Description
Include=[storage_area[...]] ¹	Backs up only the storage areas you specify.
Exclude=[storage_area[...]] ¹	Omits only the storage areas you specify.
No_Read_Only	Excludes all read-only storage areas from the backup operation.
No_Worm	Excludes all write-once storage areas from the backup operation.

¹Do not use an asterisk with the Include and Exclude qualifiers (for example, Include=* and Exclude=*).

You can specify these qualifiers when you perform either a full or an incremental backup operation. If you do not specify any of the qualifiers in Table 7–12, Oracle RCU performs a full backup operation. (See the *Oracle RCU Reference Manual* manual for complete syntax information.)

Oracle Corporation recommends that you enable after-image journaling when you perform by-area backup operations on your database. The after-image journals are crucial to ensure that you can recover all the storage areas in your database in the event of a system failure. The following table describes the outcome of a restore operation with and without after-image journaling:

If ...	Then ...
You do not have after-image journaling enabled and you restore one or more storage areas that are not current in the restored database	Oracle Rdb does not allow any transactions to use the storage areas that are not current in the restored database. In this case, you must restore the database by using the backup file from the last full backup operation of the database storage areas. Any changes made to the database since the last full and complete backup operation are not recoverable.
You have after-image journaling enabled	Using the RCU Recover command applies transactions from the after-image journal file to storage areas that are not current after the RCU Restore operation completes.

The following examples demonstrate some by-area backup operations.

Sample By-Area Backup Operations

In the following examples, assume that:

- The DEPARTMENTS storage area is a read-only storage area
- The RESUME_LISTS storage area is a write-once storage area that contains list data in the mf_personnel database

Also, note the following about the informational and warning messages returned by the Oracle RMU Backup command:

- The informational message indicates that the backup file is a partial backup file that does not include all storage areas.
- The warning message indicates that a full backup operation has never been performed on the database.

1.

```
$ RMU/BACKUP DB_DISK:[MFPERS]MF_PERSONNEL -
_$_ /EXCLUDE=(DEPARTMENTS,RESUME_LISTS) -
_$_ DBS_BACKUPS:[MFPERS]PERS_FULL_RW.RBF
%RMU-I-NOTALLARE, Not all areas will be included in this backup file
%RMU-W-NOCOMBAC, No full and complete backup was ever performed
```

This example explicitly excludes the DEPARTMENTS and the RESUME_LISTS storage areas using the Exclude qualifier. As a result only the updatable storage areas in the database are backed up. In this example, you could achieve the same result by substituting the No_Read_Only and the No_Worm qualifiers for the Exclude qualifier to back up only the updatable storage areas.

2.

```
$ RMU/BACKUP /INCLUDE=(DEPARTMENTS,RESUME_LISTS) -
_$_ DB_DISK:[MFPERS]MF_PERSONNEL -
_$_ DBS_BACKUPS:[MFPERS]PERS_FULL_RO.RBF
%RMU-I-NOTALLARE, Not all areas will be included in this backup file
%RMU-W-NOCOMBAC, No full and complete backup was ever performed
```

This example uses the Include qualifier to back up only the DEPARTMENTS read-only and RESUME_LISTS write-once storage areas. All other storage areas are excluded from the backup operations.

3.

```
$ RMU/BACKUP /INCREMENTAL=BY_AREA /INCLUDE=(RESUME_LISTS) -
_$_ DB_DISK:[MFPERS]MF_PERSONNEL -
_$_ DBS_BACKUPS:[MFPERS]PERS_FULL_RO.RBF
%RMU-I-NOTALLARE, Not all areas will be included in this backup file
```

This example shows an incremental by-area backup operation of the RESUME_LISTS write-once storage area. Oracle RMU returns a warning message whenever you perform a by-area incremental backup operation and you do not have after-image journaling enabled.

To determine if a by-area backup operation is necessary, see Section 7.8.3.2 for information about determining the last time a specific storage area was backed up.

7.10 Performing a Parallel Backup Operation

OpenVMS OpenVMS
VAX Alpha Parallel backup uses multiple, multithreaded processes to create one backup file. These processes, called executor processes, include one coordinator process and one or more worker processes that optimize the performance of the default Oracle RMU backup operation. You specify how you want to configure the executor processes in a parallel backup **plan file**. Oracle RMU reads the plan file and executes the backup operation according to the specifications therein.

Table 7–13 describes the steps you use to start a parallel backup operation:

Table 7–13 Preparing for Parallel Backup Operations

Step	Procedure	Reference						
❶	Ensure that you have installed the Oracle SQL/Services.	<i>Oracle SQL/Services Installation Guide</i> and the <i>Oracle SQL/Services Release Notes</i>						
❷	Ensure that the SQL/Services product uses the appropriate network transport. By default, the RMU Backup command uses DECnet (either DECnet for OpenVMS or DECnet/OSI) to access SQL/Services:	<i>Oracle Rdb7 Installation and Configuration Guide</i> for information about setting the <code>SQL_NETWORK_TRANSPORT_TYPE</code> configuration parameter.						
	<table border="1"> <thead> <tr> <th>If . . .</th> <th>Then . . .</th> </tr> </thead> <tbody> <tr> <td>DECnet is not available</td> <td>Oracle RMU tries to use the TCP/IP network transport.</td> </tr> <tr> <td>You want to use the TCP/IP network transport</td> <td>You can set the configuration parameter, <code>SQL_NETWORK_TRANSPORT_TYPE</code>, so that Oracle RMU tries to use the TCP/IP transport first.</td> </tr> </tbody> </table>	If . . .	Then . . .	DECnet is not available	Oracle RMU tries to use the TCP/IP network transport.	You want to use the TCP/IP network transport	You can set the configuration parameter, <code>SQL_NETWORK_TRANSPORT_TYPE</code> , so that Oracle RMU tries to use the TCP/IP transport first.	
If . . .	Then . . .							
DECnet is not available	Oracle RMU tries to use the TCP/IP network transport.							
You want to use the TCP/IP network transport	You can set the configuration parameter, <code>SQL_NETWORK_TRANSPORT_TYPE</code> , so that Oracle RMU tries to use the TCP/IP transport first.							
❸	Select the Power Utilities option when you install the Oracle Rdb software.	<i>Oracle Rdb7 Installation and Configuration Guide</i>						
❹	Create a parallel backup plan file.	Section 7.10.2						
❺	Examine and edit the plan file, if necessary, to customize the parallel backup operation.	Section 7.10.3						
❻	Execute the plan file to start a parallel backup operation.	Section 7.10.2 and Section 7.10.3 (continued on next page)						

Table 7–13 (Cont.) Preparing for Parallel Backup Operations

Step	Procedure	Reference
7	On a Windows system, use the Parallel Backup Monitor to watch the progress of the parallel backup operation.	Section 7.10.6

In addition, consider enabling tape loader synchronization to minimize operator support during a parallel backup operation.

7.10.1 Oracle RMU Commands for Parallel Backup Operations


OpenVMS Alpha  The Oracle RMU utility provides the commands and qualifiers shown in Table 7–14 to help you start a parallel backup operation.

Table 7–14 Commands and Qualifiers for a Parallel Backup Operation

Command and Qualifier	Description
RMU Backup	
[No]Execute	<p>Specifies whether or not to execute the parallel backup plan file:</p> <ul style="list-style-type: none"> Execute—Creates, verifies, and executes a backup plan file. This is the default. Noexecute—Creates and verifies the backup plan file, but it does not execute the plan. <p>You must specify the List_Plan and Parallel qualifiers when you use the [No]Execute qualifier. If you do not, Oracle RMU generates and verifies a temporary parallel backup plan, deletes the temporary plan, and returns a fatal error message.</p>
List_Plan= <i>output</i>	Specifies a name for the generated parallel backup plan file.

(continued on next page)

Table 7–14 (Cont.) Commands and Qualifiers for a Parallel Backup Operation

Command and Qualifier	Description
RMU Backup	
Parallel= Executor_Count= <i>n</i>	<p>Generates a parallel backup plan file that describes how Oracle RMU should execute the parallel backup operation. Oracle RMU executes the plan file as soon as it has been generated (unless you specify Noexecute).</p> <p>The Parallel qualifier requires the Executor_Count parameter which specifies the number of worker processes you want enabled for the parallel backup operation. The number you specify must be less than or equal to the number of tape drives you intend to use.¹</p> <p>Optionally, you can:</p> <ul style="list-style-type: none"> • Assign each worker process to a node using the Node=(<i>node_name,node_name</i>) parameter. If you specify more than one node, all nodes must be in the same cluster and the database must be accessible from all nodes in the cluster. • Gather statistics about the parallel backup operation with the Statistics parameter. You must invoke the Parallel Backup Monitor to view these statistics. <p>By default, Oracle RMU executes the plan file immediately after generating it. To generate a parallel backup plan file without executing it, include the Noexecute qualifier on the RMU Backup command line.</p>
Oracle RMU Backup Plan <i>plan-file</i>	
[No]Execute	<p>Specifies whether or not you want to execute the parallel backup plan file:</p> <ul style="list-style-type: none"> • Execute—Verifies and executes the backup plan file. This is the default. • Noexecute—Verifies the backup plan file, but it does not execute the plan. The validity check determines such things as whether the storage area names assigned to each executor exist.

¹Specify Executor_Count=1 to monitor the progress of a non-parallel backup operation using the Parallel Backup Monitor Windows interface. If you specify Executor_Count=1, Oracle RMU generates a plan file but performs the default (non-parallel) backup operation.

(continued on next page)

Table 7–14 (Cont.) Commands and Qualifiers for a Parallel Backup Operation

Command and Qualifier	Description
Oracle RMU Backup Plan <i>plan-file</i>	
List_Plan= <i>output</i>	<p>Uses an existing parallel backup plan file (specified by Oracle RMU Backup Plan) to generate and write a new plan file to the output file (specified by List_Plan). The new plan file is identical to the original plan file with the following exceptions:</p> <ul style="list-style-type: none"> • User-added comments in the original plan file are not included in the new plan file. • Formatting in the new plan file matches the standard format for Oracle RMU Backup plan files.

◆

7.10.2 Starting a Parallel Backup Operation

OpenVMS
VAX ≡≡≡

OpenVMS
Alpha ≡≡≡

Example 7–7 shows how you might use the Oracle RMU Backup commands to create and execute a parallel backup plan file.

Example 7–7 Starting a Parallel Backup Operation

```

$ REPLY/ENABLE=TAPES❶
$ RMU/BACKUP/PARALLEL=(EXECUTOR_COUNT=2❷, NODE=(NODE1,NODE2)❸) -
_ $ /LIST_PLAN=(PARALLEL.PLAN) /NOEXECUTE❹ MF_PERSONNEL MUA1:MFP.RBF, MUA2❺
.
.
.
$ RMU/BACKUP/PLAN PARALLEL.PLAN❻

```

Regarding the commands and qualifiers in Example 7–7:

- ❶ Oracle RMU sends all tape requests to the Operator; the parallel backup operation does not send tape requests to the user who entered the Backup command. Therefore, you should enter the DCL command, REPLY /ENABLE=TAPES from the operator terminal before entering the RMU Backup command.
- ❷ The Executor_Count qualifier specifies two worker (executor) processes.

- ③ The Node qualifier assigns each worker process to a different node. Before you perform a backup operation in a clustered OpenVMS systems, you must:

Note

For backup operations on multiple VMScluster nodes, you must:

- Define the logical names SQL_USERNAME and SQL_PASSWORD
 - Provide proxy access (for the user that starts the backup operation) between all nodes involved in the backup operation
-

- ④ The Noexecute qualifier prevents Oracle RMU from executing a parallel backup operation.
- ⑤ Each worker process writes to multiple tape drives.
- ⑥ The RMU Backup Plan command executes the plan file (parallel.plan) created with the RMU Backup command.

Example 7-7 creates the parallel backup plan file shown in Section 7.10.3. ♦

7.10.3 Parallel Backup Plan File

OpenVMS VAX  OpenVMS Alpha 

When a parallel backup operation executes, it reads a file called a plan file that contains information about the backup operation and how the backup operation is split across processes. When you specify the plan file, Oracle RMU executes the parallel backup operation according to the specifications in the plan.

In addition to providing information about the type of backup (online or offline, full or incremental, by-area, and so forth), the plan file contains information about each process in the plan and the work it performs. On clusters, the processes can be on different nodes in the cluster.

Oracle RMU generates a plan file automatically if you specify the List_Plan qualifier on the RMU Backup command, or you can create a plan file manually using an editor. If you use Oracle RMU to generate a list plan, you can also include the Noexecute qualifier to generate a plan file without actually performing a backup operation. Also, the Noexecute qualifier is useful for performing a validity check on the plan file. Then, you can edit the plan file before performing a parallel backup operation.

Example 7-8 shows a sample plan file.

Example 7-8 Parallel Backup Plan File

```
! Plan created on 5-JUL-1995 by RMU/BACKUP.
Plan Name = B Plan Type = BACKUP
Plan Parameters: ❶
  Database Root File = MF_PERSONNEL
  Backup File = MFP.RBF
  ! Journal = specification for journal file
  ! Tape_Expiration = dd-mmm-yyyy
  ! Active_IO = number of buffers for each tape
  ! Protection = file system protection for backup file
  ! Block_Size = bytes per tape block
  ! Density = tape density
  ! [No]Group_Size = number of blocks between XOR blocks
  ! Lock_Timeout = number of second to wait for locks
  ! Owner = identifier of owner of the backup file
  ! Page_Buffers = number of buffers to use for each storage area
NoChecksum_Verification
NOCRC
NoIncremental
Log
NoOnline
Quiet_Point
NoRewind
ACL
! [No]Scan_Optimization
! Labels = list of tape labels
End Plan Parameters

! Coordinator Process ❷
Executor Parameters :
  Executor Name = COORDINATOR_EXECUTOR
  Executor Type = Coordinator
End Executor Parameters

! Worker Processes ❸
Executor Parameters :
  Executor Name = EXECUTOR_001
  Executor Type = Worker
! Executor Node = Node name for worker
```

(continued on next page)

Example 7–8 (Cont.) Parallel Backup Plan File

```
Backup Storage Areas = ( -
MF_PERS_SEGSTR, -
DEPARTMENTS, -
EMPIDS_LOW, -
EMPIDS_MID, -
EMPIDS_OVER, -
EMP_INFO, -
JOBS, -
SALARY_HISTORY, -
RDB$SYSTEM)
Tape Drive List
Tape Drive = $111$MUA20:
Master
! Labels = list of tape labels
Tape Drive = $111$MUA22:
Master
! Labels = list of tape labels
End Tape Drive List
Executor Name = EXECUTOR_002
.
.
.
End Executor Parameters
```

- ❶ The beginning of the plan file contains the names of the root file and the backup file as well as the global parameters for the backup operation.
- ❷ This section contains information about the coordinator process. The first process described in a plan file must be the coordinator process. Currently, the only information about the coordinator process is its name, `COORDINATOR_EXECUTOR`.
- ❸ This section contains information about the two worker processes. For each of these worker processes, the plan contains the list of storage areas to back up and the names of the tape drives used by the worker process. Optionally, for cluster systems, you can specify the node on which the worker process is to run.

The master tape drive listed for the first worker process is the drive that will be the first logical volume of the backup file. All the header and database root information is written to the first volume.

When you generate a plan file with the RMU Backup command, Oracle RMU executes the plan file immediately after it is created. Also, you use the RMU Backup Plan command to execute a plan file generated with the new List_Plan qualifier of the RMU Backup command, or created manually with an editor.

Note

Do not allocate or mount any tapes manually. Oracle RMU automatically allocates and mounts tapes during the parallel backup procedure.

All worker processes coordinate with the coordinator process to get tape label names and to synchronize tape loading. After loader synchronization in a parallel backup, the first tape drive in the first worker process goes first, followed by the second tape drive for the first worker process, and so on until all tape drives for the first worker process have started. The next drive to go is the first drive for the second worker process, followed by the second drive for the second worker process, and so on.

If any worker process ends prematurely, all other worker processes stop. ♦

7.10.4 What Happens During a Parallel Backup Operation?

OpenVMS OpenVMS
VAX Alpha

The following table describes how Oracle RMU performs a parallel backup operation.

Step	Action								
❶	<p>The command you specify (either <code>RMU Backup Parallel</code> or <code>RMU Backup Plan</code>) generates a plan file, if necessary, for the parallel backup operation. The plan file is generated as follows:</p> <ul style="list-style-type: none"> Each worker process is assigned a name. For example, <code>EXECUTOR_001</code>, <code>EXECUTOR_002</code>, and so on. Each worker process is assigned a node if the plan file specifies more than one node. Nodes are assigned to each worker process, using the following algorithm: <table border="1"> <thead> <tr> <th>If ...</th> <th>Then ...</th> </tr> </thead> <tbody> <tr> <td>You specify the same number of nodes as there are worker processes</td> <td>Each worker process is assigned a node. If you specify three nodes and three worker processes, the first node that appears in your node name list is assigned to the first worker process, the second node in the list is assigned to the second worker process, and so on.</td> </tr> <tr> <td>You specify fewer nodes than worker processes</td> <td>Nodes are assigned using a round-robin approach. For example, if you specify <code>Nodes=(ONE,TWO)</code> and <code>Executor_Count=3</code>, then <code>EXECUTOR_001</code> is assigned to node ONE, <code>EXECUTOR_002</code> is assigned to node TWO, and <code>EXECUTOR_003</code> is assigned to node ONE.</td> </tr> <tr> <td>You do not specify a list of nodes</td> <td>The backup operation uses the node from which the command is entered. Nodes are assigned to the worker processes using the round-robin approach described previously.</td> </tr> </tbody> </table> <ul style="list-style-type: none"> Each tape drive is assigned to a worker process. <ul style="list-style-type: none"> Each tape drive is assigned to a worker process using the round-robin approach. If you specify three tape drives and three worker processes, the first tape drive that you specify is assigned to the first worker process, the second tape drive to the second worker process, and so on. The number of tape drives must match the number of worker processes. If the tape drives are not accessible from all nodes in your cluster, use the algorithm (described in the table in step 4) for assigning nodes and tape drives to worker processes. Ensure that nodes and tape drives will be correctly matched when the coordinator process assigns them to each worker process. Alternatively, you can specify the <code>Noexecute</code> qualifier and edit the plan file to reassign the tape drives and nodes appropriately. Then execute the plan file using the <code>RMU Backup Plan</code> command. Each storage area (partition) to be backed up is assigned to a worker process. <ul style="list-style-type: none"> Oracle RMU sorts the storage areas to be backed up by size, and then assigns them, in a round-robin manner, to each worker process. The root file is always backed up by the first worker process. All areas in the database are assigned to a worker process unless you use the <code>Include</code>, <code>Exclude</code>, <code>No_Worm</code>, or <code>No_Read_Only</code> qualifiers. 	If ...	Then ...	You specify the same number of nodes as there are worker processes	Each worker process is assigned a node. If you specify three nodes and three worker processes, the first node that appears in your node name list is assigned to the first worker process, the second node in the list is assigned to the second worker process, and so on.	You specify fewer nodes than worker processes	Nodes are assigned using a round-robin approach. For example, if you specify <code>Nodes=(ONE,TWO)</code> and <code>Executor_Count=3</code> , then <code>EXECUTOR_001</code> is assigned to node ONE, <code>EXECUTOR_002</code> is assigned to node TWO, and <code>EXECUTOR_003</code> is assigned to node ONE.	You do not specify a list of nodes	The backup operation uses the node from which the command is entered. Nodes are assigned to the worker processes using the round-robin approach described previously.
If ...	Then ...								
You specify the same number of nodes as there are worker processes	Each worker process is assigned a node. If you specify three nodes and three worker processes, the first node that appears in your node name list is assigned to the first worker process, the second node in the list is assigned to the second worker process, and so on.								
You specify fewer nodes than worker processes	Nodes are assigned using a round-robin approach. For example, if you specify <code>Nodes=(ONE,TWO)</code> and <code>Executor_Count=3</code> , then <code>EXECUTOR_001</code> is assigned to node ONE, <code>EXECUTOR_002</code> is assigned to node TWO, and <code>EXECUTOR_003</code> is assigned to node ONE.								
You do not specify a list of nodes	The backup operation uses the node from which the command is entered. Nodes are assigned to the worker processes using the round-robin approach described previously.								
❷	<p>Oracle RMU verifies the generated plan file, including ensuring that the nodes, tape drives, and storage area names for the specified database exist.</p>								

Step	Action
③	<p>Oracle RMU executes the plan file (unless you include the Noexecute qualifier) as follows:</p> <ol style="list-style-type: none"> 1. Your process notifies Oracle SQL/Services that a parallel backup operation is about to be performed. 2. Using the RMU\$SRV70 account¹, Oracle SQL/Services creates a coordinator process that serves as the backup operation execution manager, and creates the number of worker processes specified in the plan file. The coordinator and worker processes change identities to run as the user executing the parallel backup operations. 3. Each worker process requests a tape label from the coordinator process for the first tape label, and makes additional requests for tape labels each time a worker process requires a new tape and tape label. 4. Oracle RMU allocates and mounts each tape. 5. The first worker process backs up the database root file before any other worker processes can begin. 6. Once the database root file is backed up, each worker process backs up its assigned storage areas (as defined in the plan file) using its assigned tape drives.
④	<p>Oracle RMU dismounts and unloads all the tapes when the backup operation is complete.</p>

¹The RMU\$SRV70 account is created automatically when you install the Oracle SQL/Services software. The RMU\$SRV70 account is required for parallel backup operations. Refer to the *Oracle SQL/Services Release Notes* for more information.

7.10.5 Using Loader Synchronization When Performing a Parallel Backup Operation

OpenVMS OpenVMS
VAX Alpha The Loader_Synchronization qualifier allows you to preload tapes and preserve tape order to minimize the need for operator support. The following example demonstrates how the backup operation proceeds when you specify the Loader_Synchronization qualifier and the Parallel qualifier with the Oracle RMU Backup command.

The following list describes how Oracle RMU synchronizes tape loading and labeling. The backup scenario described includes three parallel worker processes, and each worker process is assigned three tape drives:

1. EXECUTOR_001 begins backing up to its three tape drives in succession, using tape labels 001, 002, and 003.
2. Concurrently, EXECUTOR_002 begins backing up to its three tape drives in succession, using tape labels 004, 005, and 006.
3. Concurrently, EXECUTOR_003 begins backing up to its three tape drives in succession, using tape labels 007, 008, and 009.

4. Assume EXECUTOR_002 fills one tape on each drive first. It reports this status to the coordinator process.
5. The coordinator process directs EXECUTOR_002 to wait. Likewise, when EXECUTOR_003 fills its first three tapes, the coordinator directs it to wait, and so on.
6. The coordinator process allows each worker process to resume backup operations *only* when each executor has filled its first three tapes.

This procedure ensures that you can correctly match the tape labels to the backup tapes.

See Section 7.13.2.3 for additional information about the use of the Loader_Synchronization qualifier. ♦

7.10.6 Monitoring the Progress of a Parallel Backup Operation

OpenVMS VAX  OpenVMS Alpha 

If you enabled the Statistics option when you entered the RMU Backup Parallel command, you can monitor the progress of the parallel backup operation on your Windows system using the Parallel Backup Monitor. The Parallel Backup Monitor is a Windowing interface that runs on Windows 3.1, Windows 95, Windows Intel Alpha 3.51, and Windows Intel NT 3.51.

Note

You can use the Parallel Backup Monitor Windows interface to monitor the progress of the default (non-parallel) backup operation. Do this by starting the backup operation with the RMU Backup Parallel=Executor_Count=1 Statistics command. Oracle RMU generates a plan file that specifies one worker process, and gathers statistics as it performs a default (non-parallel) backup operation. Then, you can monitor the progress of the operation using the Parallel Backup Monitor.

See the *Oracle SQL/Services Installation Guide* and the *Oracle SQL/Services Release Notes* for more information about installing and using SQL/Services. Also, see Windows help for information about using the Parallel Backup Monitor. ♦

7.11 Performing Online Backup Operations

Oracle RMAN provides the capability to perform online backup operations. This is essential in situations where the database must remain available to users, and it is unacceptable to take the database off line to perform maintenance operations.

You can perform either a full, incremental, or parallel backup operation while your database is on line. You do not need to close the database or deny access to users in order to perform an Oracle RMAN backup operation. The Oracle RMAN online backup operation permits concurrency with all other types of transactions except read/write exclusive or batch-update transactions.

Concurrent access is made possible through the use of snapshot files. By using the snapshot files, the online Oracle RMAN backup operation lets you back up a consistent view of the database while allowing users to continue update operations. The following list describes what happens during an online backup operation:

- Oracle RMAN starts a read-only transaction on the database to ensure that it sees the data records as they were at the time the transaction was started. Snapshots ensure that read-only transactions can see a consistent view of the data while update transactions modify records. The read/write exclusive and batch-update transactions are not allowed because neither transaction type causes before-images of updated rows or records to be written to snapshot files.
- Oracle RMAN takes out a quiet-point lock and waits for all active read/write transactions to complete; the online Oracle RMAN backup operation proceeds when the database reaches a **quiet point** (a moment when there are no active read/write transactions). The benefit of waiting for the quiet point is that there can be no lock conflicts or deadlocks that may prevent the backup operation from proceeding. Section 7.11.1 describes quiet-point locks in more detail.

Table 7–15 describes how to start an online backup operation.

Table 7–15 Starting an Online Backup Operation

Procedure	Description
Determine if snapshot files are enabled	<p data-bbox="305 688 1341 762">To determine if the database has snapshot files enabled or disabled, use the RMU Dump command with the Header qualifier, as shown in the following example, and inspect the snapshot parameter setting for any storage area.</p> <pre data-bbox="305 783 943 972">\$ RMU/DUMP/HEADER DB_DISK:[MFPERS]MF_PERSONNEL.RDB . . . Snapshots... - Snapshots are disabled - Snapshot area ID number is 11</pre> <p data-bbox="305 989 1341 1031">If snapshot files are disabled when you issue an RMU Backup Online command, Oracle RMU returns an error.</p>

(continued on next page)

Table 7–15 (Cont.) Starting an Online Backup Operation

Procedure	Description
Enable snapshot files	<p>Snapshot files make it possible to perform Oracle RMU backup operations while the database is on line and accessible to users. If snapshot files are disabled, change the setting to enable snapshots:</p> <ol style="list-style-type: none"> 1. Close the database to restrict access. 2. Enter the SQL ALTER DATABASE statement, as shown in the following example: <pre> \$ SQL = "\$SQL\$" \$ \$ -- Use ALTER DATABASE statement to set OPEN IS MANUAL. \$ SQL SQL> ALTER DATABASE FILENAME DB_DISK:[MFPERS]MF_PERSONNEL cont> OPEN IS MANUAL; SQL> EXIT \$! Determine how many users are attached to the database; then \$! determine the best way to close the database - let users \$! complete their transactions first, then close the database. \$ \$ RMU/DUMP USERS DB_DISK:[MFPERS]MF_PERSONNEL \$ \$! Close the database by using a Noabort qualifier. \$ \$ RMU/CLOSE DB_DISK:[MFPERS]MF_PERSONNEL /NOABORT /CLUSTER \$ \$! Check to see that all transactions are complete and no users \$! are attached to the database. \$ \$ RMU/DUMP USERS DB_DISK:[MFPERS]MF_PERSONNEL \$ \$! Enable .snp files. \$ \$ SQL SQL> ALTER DATABASE FILENAME DB_DISK:[MFPERS]MF_PERSONNEL cont> SNAPSHOT IS ENABLED; SQL> EXIT \$ \$! Open the database by entering RMU Open command. \$ \$ RMU/OPEN DB_DISK:[MFPERS]MF_PERSONNEL </pre>
Start the backup operation	<p>After you enable snapshot files and open the database again, start an online backup operation by including the Online qualifier on the RMU Backup command line:</p> <pre> \$ RMU/BACKUP/ONLINE DB_DISK:[MFPERS]MF_PERSONNEL.RDB - _ \$ DBS_BACKUPS:[MFPERS]PERS_FULL_FEB22.RBF </pre>

After the online Oracle RMU backup operation has started, users can invoke the database and begin new transactions, as long as they do not require exclusive access to the database, a table or view, or an index structure that is currently being backed up.

7.11.1 Avoiding Lock Conflicts

By default, an online backup operation acquires a quiet-point lock when the operation begins. This causes Oracle RMU to wait for all active transactions to complete before the backup operation begins. The following sections describe using the `Quiet_Point` and `Noquiet_Point` qualifiers.

Acquiring Quiet Point Locks

An online backup operation acquires the quiet-point lock for a time when the backup operation begins. This can cause a brief interruption in the database activity, so you should schedule its use accordingly.

When Oracle RMU attempts to acquire the quiet-point lock, it inhibits the initiation of new database transactions until the lock is released. Oracle RMU acquires the lock upon completion of the update transactions that were active when the lock was requested. The lock is held for only a few seconds, just long enough for the backup operation to acquire a consistent image of the database root file internal structures.

Section 7.11.2 describes how to control the interval of time that Oracle RMU waits for a quiet point.

Using the `Noquiet_Point` Qualifier

As another option, you can specify the no-quiet-point online backup operation if you are sure there are no lock conflicts between the online backup operation and your concurrently running application.

Use the `Noquiet_Point` qualifier so that Oracle RMU can proceed with the backup operation as soon as you enter the `Backup` command. However, you must make certain that there are no active transactions using exclusive locks.

Note

If you specify the `Noquiet_Point` qualifier, Oracle RMU takes out a lock but there is a risk of lock conflict. If Oracle RMU cannot acquire concurrent-read locks on all physical and logical areas, the online backup operation fails with a lock conflict. Also, you must keep the after-image journal files from the last quiet-point backup operation or offline backup operation.

Quiet_Point and Noquiet_Point Trade-Offs

As you devise an overall backup strategy for your database and after-image journal files, you might want to consider using a combination of Quiet_Point and Noquiet_Point qualifiers.

For example, although a noquiet-point backup operation is faster than a quiet-point backup operation, it usually results in a longer recovery operation. This is because transactions can span after-image journal files when you specify the RMU Backup After_Journal Noquiet_Point command. Thus, you might have to apply numerous .aij files to recover the database. In a worst-case scenario, this could extend back to your last quiet-point after-image journal or database backup operation. If you rarely perform quiet-point backup operations, recovery time could be excessive.

To balance these trade-offs, consider performing regularly scheduled quiet-point after-image journal backup operations followed by noquiet-point database backup operations. (Conversely, you could perform a quiet-point backup operation on the database followed by noquiet-point backup operation on the after-image journal. However, a quiet-point backup of the after-image journals usually takes less time than a quiet-point backup of the database.) Periodically performing a quiet-point after-image journal backup operation helps to ensure that your recovery time does not become excessive.

7.11.2 Setting Lock Timeout Intervals

If you do not want an online backup operation to wait indefinitely to acquire the quiet-point lock, you can control how long the RMU backup operation waits by including the Lock_Timeout qualifier on the RMU Backup command.

For example, suppose it is important that your database be incrementally backed up every 12 hours, and from experience you notice that this event usually begins within 5 minutes of the designated start time and requires about 1 hour to complete. A command procedure automatically starts this online incremental backup operation twice each day. If a quiet-point lock cannot be acquired and the backup operation does not begin within 5 minutes, you want to be notified so you can investigate further and decide the best course of action to take.

To do this, set the lock timeout interval to 300 seconds, as shown in Example 7-9, and then have your command procedure send you mail notifying you that either the backup operation was successful or it never started because the lock timeout interval was exceeded.

Example 7–9 Setting the Lock Timeout Interval to 300 Seconds

```
$ RMU/BACKUP/ONLINE/INCREMENTAL/LOG /LOCK_TIMEOUT=300 -
_ $ DB_DISK:[MFPERS]MF_PERSONNEL.RDB -
_ $ DBS_BACKUPS:[MFPERS]PERS_INC_FEB7_1200.RBF
```

Example 7–10 Starting an Offline Backup Operation When a User Is Attached to the Database

```
$ RMU/BACKUP DB_DISK:[MFPERS]MF_PERSONNEL.RDB -
_ $ DBS_BACKUPS:[MFPERS]PERS_FULL.RBF
%RMU-F-FILACCERR, error opening database root file
    DB_DISK:[MFPERS]MF_PERSONNEL.RDB;1
-SYSTEM-W-ACCONFLICT, file access conflict
```

The Nonline qualifier is the default for the RMU Backup command. If you try to issue a backup command against an attached database, (that is, a database in which one or more users have invoked the SQL ATTACH statement) without specifying the Online qualifier, you receive the error message shown in Example 7–10.

7.12 Backing Up a Database to a Disk Device

Although Oracle RMU is optimized to perform Oracle Rdb database backup operations to tape, there may be circumstances where you want to back up to disk.

If you back up an Oracle Rdb database to a disk device, remember to locate the database root file and the backup file on separate disks. If the disk device where the database root file resides suffers a hardware failure (disk head failure, for instance), the backup file is useless if it too resides on the corrupt disk. You do not want to risk losing both your database and your backup files if you lose a disk.

Disk Backup on OpenVMS Systems

OpenVMS OpenVMS
VAX Alpha

If you choose to perform an Oracle RMU backup operation to disk, estimate the size of the backup file and set the OpenVMS Record Management System (RMS) to have a large extent size. Try to size the extents so that the entire backup file fits in a few extents while not wasting disk space.

For example, find the size of your most recent full backup file:

```
$ DIRECTORY/SIZE=ALL FULL_BACKUP_MONDAY.RBF
Directory DISK$A:[ORACLEUSER]

FULL_BACKUP_MONDAY.RBF:1 1449/1449
Total of 1 file, 1449/1449 blocks.
```

Then, set the extent size so that the backup file fits in two extents:

```
$ SET RMS_DEFAULT/EXTENT=750
```



7.13 Backing Up a Database to Tape Devices

Oracle RMU is optimized for backing up Oracle Rdb databases to tape media. Oracle RMU support for multiple output threads and its ability to balance I/O across all storage areas result in the best possible performance for database backup.

There are several ways of backing up directly to tape:

- Back up the database to one tape drive, onto one or more tape volumes loaded sequentially.
- Back up the database to multiple tape drives, onto multiple tape volumes loaded concurrently among all master drives and loaded concurrently among all slave drives but sequentially between master and slave drives.
- Back up the database to multiple tape drives, onto multiple tape volumes loaded concurrently among all tape drives.

In general, you should use a single tape drive for backing up small databases that may all fit on one tape or when there is only one tape drive on the system. Use multiple tape drives and tape controllers for backing up moderate to very large databases and when two or more tape controllers and two or more tape drives are part of each tape controller in the system.

Note

Do not use more than a total of 17 characters in the file name and extension. Otherwise, the file name is truncated to the first 17 characters and is stored without the .rbf extension.

7.13.1 Mounting One or More Tapes on a Single Tape Drive

When your configuration has only one tape drive, you can back up a database directly to one or more tapes on that drive by using the following procedure:

Step	Action
❶	†Allocate the tape drive.
❷	†Mount the tape as a foreign volume.
❸	Perform the backup operation.
❹	If you are using only one tape, rewind the tape to prepare for verifying the backup file. Ignore this step if you are not verifying the backup file.
❺	If you need to mount two or more tapes on the tape drive, dismount the last tape and mount the first tape to prepare for verifying the backup file. Ignore this step if you are not verifying the backup file.
❻	If you experience hardware or media errors during the backup operation, verify the backup file. The RMU Dump Backup_File command detects media errors, but it does not verify contents of the backup.
❼	†Dismount the tape.
❽	†Deallocate the tape drive.

†Applies to OpenVMS systems that do not use parallel backup operations.

Example 7–11 shows the procedure to back up the mf_personnel database to one tape.

Example 7–11 Performing a Full Backup Operation to One Tape

```
$ RMU/VERIFY DB_DISK:[MFPERS]MF_PERSONNEL
$
$ ALLOCATE $111$MUA0:
$ MOUNT/FOREIGN $111$MUA0:
$ RMU/BACKUP /LOG /REWIND /DENSITY=6250 /LABEL=BACK01 -
_ $ DB_DISK[MFPERS]MF_PERSONNEL.RDB $111$MUA0:PERS_FULL.RBF
$
$ ! The Rewind qualifier rewinds the tape to the beginning and
$ ! initializes the tape at 6250 bits per inch as specified
$ ! by the Density=6250 qualifier and gives it a label of BACK01 as
$ ! specified in the Label qualifier. Then the backup file named
$ ! pers_full.rbf is created and written to this output volume.
$ ! Note: If the label on the tape does not match the label specified
$ ! in the Label qualifier, you receive a message and are asked what
$ ! course of action to follow. See Section 7.13.4
$ ! for information about checking tape labeling.
```

(continued on next page)

Example 7–11 (Cont.) Performing a Full Backup Operation to One Tape

```
$
$ ! If you want to check the tape for media errors, enter the next two
$ ! commands to rewind the tape and begin media error detection.
$ ! Otherwise, skip to the next step to dismount the tape.
$
$ SET MAGTAPE/REWIND
$ RMU/DUMP/BACKUP_FILE $111$MUA0:PERS_FULL.RBF
$
$ ! When the operating system default prompt returns and no error messages
$ ! are displayed, verification is complete.
$
$ DISMOUNT $111$MUA0:
$ DEALLOCATE $111$MUA0:
```

In Example 7–11 a single tape drive \$111\$MUA0: is used. The Rewind, Density=6250, and Label=BACK01 qualifiers indicate that the tape is to be rewound to the beginning, initialized at 6250 bits per inch and given a label of BACK01; then, the backup file named pers_full.rbf is created and written to the tape. If the volume label disagrees with the label specified by the Label qualifier or by the default (the backup file name), an informational message displays.

Caution

If you back up two or more databases in succession to the same tape, do not use the Rewind qualifier in your second and subsequent RMU Backup commands. This qualifier rewinds and reinitializes the tape, destroying all existing backup files on the tape. Specify the Norewind qualifier or none at all (the default is Norewind) to ensure that the next database backup file created starts at the current end-of-tape (EOT).

If you have a large database, you may need more than one tape to hold the database backup file. In this case, you are prompted to mount additional tapes as each fills up until the backup procedure is completed. (If you run the backup operation as a batch job, Oracle RMU notifies the operator using an OPCOM message.)

You must dismount the last tape and mount the first tape again to verify the backup file. Also, when you verify the backup file, you are requested to mount each tape in sequence to complete the RMU Dump Backup_File operation. Using two tapes for a backup operation is shown in Example 7–12.

Example 7–12 Performing a Full Backup Operation to Multiple Tapes Mounted on a Single Tape Drive

```
$ RMU/VERIFY DB_DISK:[MFPERS]MF_PERSONNEL
$
$ ALLOCATE $111$MUA0:
$
$ MOUNT/FOREIGN $111$MUA0:
$ RMU/BACKUP /LOG /REWIND /DENSITY=6250 /LABEL=(BACK01,BACK02) -
_$ DB_DISK[MFPERS]MF_PERSONNEL.RDB $111$MUA0:PERS_FULL.RBF
$
$ ! Mount the next volume, physically remove BACK01, and place another
$ ! unused volume on the tape drive. This volume is given the label
$ ! BACK02.
$ ! Note: If the label on the tape does not match the label specified
$ ! in the Label qualifier, you receive a message and are asked what
$ ! course of action to follow. See Section 7.13.4
$ ! about checking tape labeling for more information.
$
$ ! Dismount the second tape in order to mount the first tape again
$ ! in preparation for verifying the backup file. Dismount the first tape
$ ! and mount the second tape when prompted to do so. Ignore these steps
$ ! if you are skipping the next step.
$ !
$ DISMOUNT $111$MUA0:
$ MOUNT/FOREIGN $111$MUA0:
$
$ ! Mount the first volume, BACK01. Then mount the next volume BACK02
$ ! in sequence as requested. When the operating system default prompt
$ ! returns and no error messages are displayed, verification is complete.
$ !
$ RMU/DUMP/BACKUP_FILE $111$MUA0:PERS_FULL.RBF
$
$ DISMOUNT $111$MUA0:
$ DEALLOCATE $111$MUA0:
```

As the backup operation proceeds, be sure to mark each tape with the tape label after it is dismounted. In addition, you should mark the first tape with the date and time the backup file was created.

7.13.2 Using Multiple Tape Drives

If your system is configured with two or more tape drives, you can back up a database directly to two or more tapes on each tape drive with the following procedure:

Step	Action
❶	†Allocate each tape drive.
❷	†Mount the first tape on each drive.
❸	Perform the backup operation.
❹	Ignore this step if you are not verifying the backup file. If you need to mount two or more tapes sequentially on the same tape drive, dismount the last tape and mount the first tape on each tape drive.
❺	If you experience hardware or media errors during the backup operation, verify the backup file. The RMU Dump Backup_File command detects media errors, but it does not verify save-set contents.
❻	†Dismount the last tape.
❼	†Deallocate each tape drive.

†OpenVMS only.

This multiple-step procedure is shown in Example 7–13.

Example 7–13 Performing a Full Backup Operation to Multiple Tapes Mounted on Multiple Tape Drives

```

$ RMU/VERIFY DB_DISK:[MFPERS]MF_PERSONNEL
$ ALLOCATE $111$MUA0:
$ ALLOCATE $222$MUA1:
$ MOUNT/FOREIGN $111$MUA0:
$ RMU/BACKUP /LOG /REWIND /DENSITY=6250 /LABEL=(BACK01,BACK02) -
_ $ DB_DISK:[MFPERS]MF_PERSONNEL.RDB -
_ $ $111$MUA0:PERS_FULL.RBF,$222$MUA1:

$ ! Note: If the label on the tape does not match the label specified
$ ! in the Label qualifier, you receive a message and are asked what
$ ! course of action to follow. See Section 7.13.4
$ ! about tape labeling for more information.
$
$ DISMOUNT $111$MUA0:,$222$MUA1:
$ MOUNT/FOREIGN $111$MUA0:

```

(continued on next page)

Example 7–13 (Cont.) Performing a Full Backup Operation to Multiple Tapes Mounted on Multiple Tape Drives

```
$ !
$ ! Ignore these steps if you are skipping the next step.
$ ! Dismount the second tape on each tape drive in which there is more
$ ! than one tape in order to mount the first tape again in preparation
$ ! for verifying the backup file.
$ !
$ RMU/DUMP/BACKUP_FILE $111$MUA0:PERS_FULL.RBF,$222$MUA1:
$ DISMOUNT $111$MUA0:,$222$MUA1:
$ DEALLOCATE $111$MUA0:
$ DEALLOCATE $222$MUA1:
$ !
$ ! Dismount the first tape and mount the second tape when prompted to do so.
$ !
```

In Example 7–13, the database fits on two tapes. Because each tape is configured to a different drive, each tape is written to simultaneously in a multithreaded backup operation, and each is seen as a master tape drive. This is the preferred backup method for very large databases. If more than one tape is required per tape drive, Oracle RMU generates labels (or you can specify all labels in the Label qualifier), in sequence. As each tape fills up, Oracle RMU prompts you to mount the next volume in sequence. If both tape devices are configured to the same drive, a master/slave arrangement is assumed and each tape is written to in sequence, not simultaneously.

7.13.2.1 Using Concurrent Tape Drives Efficiently

If you are using two or more tape drives, Oracle RMU attempts to determine an efficient concurrent use of the drives. To this end, Oracle RMU assigns some of the drives as masters and some as slaves. Oracle RMU is designed to optimize backup performance and automatically load balance the I/O operations. However, Oracle RMU provides the Master, Loader_Synchronization, and Journal qualifiers for the Backup command to manage tape labeling and assignments.

In addition, for very large backup operations requiring many tape volumes, you might consider purchasing one of the storage library or archiving products available. These product are available to automatically manage tape labeling for you.

The following sections describe the Master, Loader_Synchronization, and Journal qualifiers.

7.13.2.2 Controlling Tape Concurrency with the Master Qualifier

When you have several tape control units (TCUs), each of which contains several tape drives, you should use the Master qualifier to explicitly assign which tape drives are master tapes. Oracle RMU associates each master tape drive with a particular backup output thread. This strategy not only gives you more control over the level of tape concurrency, it also eliminates any uncertainty about which tape drives are masters and which are slaves.

Note

The output performance of the backup operation decreases considerably if you use the Master qualifier on a tape drive that is not a master tape drive.

Drives that are not specified as master drives are treated as slave drives. If you specify all drives as master drives, the Oracle RMU backup operation will write to all drives concurrently.

Oracle Corporation recommends that you use the Master qualifier in conjunction with the Loader_Synchronization qualifier (described in Section 7.13.2.3). If you specify the Master qualifier without also specifying the Loader_Synchronization qualifier on sets of tape drives, each master and slave set of tape drives will operate independently of other master and slave sets. For example, suppose you have two tape control units, TCU-A and TCU-B:

- TCU-A has drive1 and drive2
- TCU-B has drive3 and drive4.

If you specify drive1 and drive3 as master tape drives, then Oracle RMU begins writing to drive1 and drive3 concurrently, while drive2 and drive4 remain idle. When the volume on drive1 fills, Oracle RMU switches over and writes the next volume on drive2. Similarly, when a volume fills on drive3, Oracle RMU writes the next volume on drive4. When a volume on drive2 or drive4 fills, Oracle RMU begins a new volume on drive1 or drive3, respectively.

7.13.2.3 Preloading Tapes Using Load Synchronization

For large backup operations, you can minimize the need for operator support by using the `Loader_Synchronization` qualifier to preload tapes into loaders or stackers for concurrent tape drive operation. Loader synchronization ensures that the tape order can be preserved in the event that a restore operation from these tapes becomes necessary.

When you use the `Loader_Synchronization` qualifier, Oracle RMU labels the volumes following the order of tape drives specified in the RMU Backup command. During the backup operation, Oracle RMU writes to the first set of tape volumes concurrently then waits until each tape in the set is finished before assigning the next set of tape volumes. That is, rather than mounting the next tape on the first drive that becomes idle, Oracle RMU waits until all the drives become idle and mounts all the tapes at once. This ensures that the tapes can be loaded into the loaders or stackers in the order required by the restore operation as specified with the `Label` qualifier.

Use of the `Master` qualifier with the `Loader_Synchronization` qualifier is meaningful when you are using several tape control units (TCUs), each of which contains several tape drives. For example, suppose you have two tape control units, TCU-A and TCU-B:

- TCU-A has drive1 and drive2
- TCU-B has drive3 and drive4

If you specify all four drives as masters, the RMU backup operation will write to all four drives concurrently. If you specify drive1 and drive3 as master drives, they will be written to concurrently and drive2 and drive4 will remain idle until both drive1 and drive3 have finished. The switch to drive2 and drive4 will occur concurrently when tapes on drive1 and drive3 are both full.

Use of the `Loader_Synchronization` qualifier can result in reduced performance. Note the following about backup performance when you use loader synchronization:

- To enhance backup performance, do not let any drive remain idle. The operator should place the next identified volume on the first drive that becomes idle. However, because the order in which the drives become idle depends on many uncontrollable factors and cannot be predetermined, the drives cannot be preloaded with tapes.
- Because the cost of using the `Loader_Synchronization` qualifier is dependent on the hardware configuration and the system load, the cost is unpredictable. A 5% to 20% additional elapsed time for the operation is typical. You must determine if the benefit of a lower level

of operator support compensates for the loss of performance. The `Loader_Synchronization` qualifier is most useful for large backup operations.

- The following case might lead to unexpected results in tape drive utilization:

When you specify the `RMU Backup` command with the `Loader_Synchronization` qualifier, Oracle RMU tries to determine the load and divide it evenly (based on storage area size) among the tape drives prior to actually beginning to write to tape. However, suppose that one large storage area has very little data in it. Furthermore, assume that you have allocated three tape drives. Because Oracle RMU predetermines the order in which the storage areas are backed up to tape, it could result in an uneven distribution of write I/Os (for example, only one storage area being written on the first drive, three to the second drive, and three to the third drive) or some drives might remain idle unexpectedly. Moreover, the second and third drives will not have the labels that you expect.

- For very large backup operations requiring many tape volumes, managing the physical marking of tape volumes can be difficult. In such a case, you might consider purchasing and using a storage library system or archiving software that automatically manages tape labeling for you.

Table 7–16 describes implicit and explicit Oracle RMU tape handling and labeling.

Table 7–16 Implicit and Explicit Tape Labeling

If You Use . . .	Then . . .
Explicit labeling	<p>You must specify the complete list of volume labels in the <code>RMU Backup</code> command line.</p> <p>In this instance, the <code>Loader_Synchronization</code> qualifier is required. One disadvantage with using the <code>Loader_Synchronization</code> qualifier is that because not all tape threads back up equal volumes of data, some operator support is needed to load the tapes in stages as backup threads become inactive. If a mistake is made in loading the tape volume, Oracle RMU will prompt the operator for tape disposition, as described in Section 7.13.4.</p>
Implicit labeling	<p>Oracle RMU labels the volumes by using the sequentially ordered labels such as <code>TAPE00</code>, <code>TAPE01</code>, and so forth. The tape volumes can be loaded in any order by the operator. Oracle RMU assigns labels to the tape volumes in the order in which they are written.</p>

When you use the `Loader_Synchronization` qualifier, you need to be careful about physically marking your tapes as they come off the tape drives to ensure that they are correctly labeled and to maintain the proper order for the restore operation. For this reason, you should use the `Journal` qualifier to create a

journal file that records a description of the backup operation. (The Journal qualifier is described in Section 7.13.2.4.)

Also, see Section 7.10 for information on how the Loader_Synchronization and Parallel qualifiers interact.

7.13.2.4 Optimizing Tape Utilization Using a Journal File

To improve tape performance for RMU Restore and RMU Dump Backup_File operations that must read multiple tapes, you can use the Journal qualifier with the RMU Backup command. Having a journal file facilitates a restore operation or dump backup operation when these are necessary.

When you specify the Journal qualifier, the RMU Backup command creates a journal (.jnl) file that describes the backup operation including identification of the tape volumes, their contents, and the order in which the labels were written to each volume. You must write the journal file to disk; it cannot be written to tape along with the backup file.

Oracle Corporation recommends you use the Journal qualifier. It is particularly useful when you employ implicit labeling (allow Oracle RMU to generate tape labels) in a multidrive, unattended backup operation, because the journal records the order in which the labels are written.

7.13.3 Avoiding Underrun Errors Using Cyclic Redundancy Checks

An underrun error occurs if the data being written fails to arrive at the tape drive when the drive is ready for it. The usual reason for HSC tape underrun errors is CI overload (queuing delays) or CI error recovery. When an underrun error occurs, the tape is spinning and the drive is writing. The tape cannot stop instantaneously. Instead, it either stops as soon as it can or continues writing nonsense (say zeros) until the write request is finished. The HSC drives write zeros to fill out the block being written (to the requested length) including writing the correct parity and check characters.

Not all tape drives or system configurations and loadings generate underrun errors. In particular, tape drives with cache generally cannot generate underrun errors. Underrun errors may indicate that the system is either broken or seriously undersized and should be fixed or properly sized for the task.

Automatic Error Correction

The RMU Backup, OpenVMS BACKUP, and Digital UNIX `tar` commands operate on the basis that media errors are more common than underrun errors. To recover from errors, these commands leave the incomplete record and rewrite it in a different tape position. The incomplete record does not generate a correct cyclic redundancy check (CRC), so if you specify the RMU Backup command with the `Crc` qualifier, Oracle RMU detects the incomplete record and uses the good duplicate record; thus, the original error is corrected. If the duplicate records are also bad, you can reconstruct the record using XOR error recovery. If you do not use XOR error recovery, you cannot restore the database with the backup file on that tape if both the record and its duplicate records are lost.

When Is the Crc Qualifier Required?

If you do not use the `Crc` qualifier, the incomplete record is read. The initial portion of the incomplete record is correct and passes all the available tests, so the record is used and the subsequent duplicate records are ignored. This happens because the incomplete record is read by the tape drive as a complete record of the correct length but the tail end of it is replaced by zeros. The tape drive does not detect an error when the record is read because it wrote the correct parity and check characters to the tape. Therefore, processing the incomplete record can produce problems. The `Crc` qualifier is required if underrun errors are possible.

Crc Qualifier Options

The `Crc` qualifier has several options that, when specified, append a 32-bit end-to-end error detection code value to each block of the backup file. This provides added data integrity for devices and system data paths with less reliable error checking. The `Crc` qualifier options are shown in the following table:

Qualifier	Description
<code>Crc [=Autodin_II]</code>	Uses the AUTODIN-II polynomial. The default for NRZ/PE (800/1600 bits/inch) tape drives. The computation uses the hardware instruction if it is available and the software emulation of CRC on processors that do not implement the hardware CRC instruction (for example, VAX 6000). It is the most reliable end-to-end error detection provided (probability of a missed error is less than one in one billion).

Qualifier	Description
Crc=Checksum	<p>Uses addition with an end-around carry.</p> <p>The default for GCR (6250 bits/inch) tape drives and TA78, TA79 and TA81 tape drives. This is the same computation used to check the checksum on the database pages on disk. These HSC drives have adequate error detection capability, but CI contention may cause data underruns and unrecoverable restore errors. This qualifier option has a modest end-to-end error detection capability (probability of a missed error is less than one in one thousand); it is more than adequate for detecting the data underrun errors, and it is at least six times faster than the hardware CRC computation.</p>
Nocrc	<p>Disables end-to-end error detection.</p> <p>The default for TA90 (IBM 3480 class drives).</p>

The overall effect of these defaults is to make tape reliability comparable to disk reliability. Use the guidelines in the following table for qualifier usage:

If You Expect to Retain Your Backup Tapes . . .	Then . . .
Longer than 1 year	The Nocrc qualifier default may not be adequate. In this case, use the Crc=Checksum qualifier.
Longer than 3 years	Always use the Crc=Autodin_II qualifier.

Never retain backup tapes longer than five years without copying the data to fresh media. This ensures against tape deterioration with age.

7.13.4 Checking Tape Labels

This section describes the series of steps Oracle RMU performs to check labels on backup tapes. RMU processes tape labels to ensure that incorrect tape volumes are not used to back up Oracle Rdb databases.

Oracle RMU performs automatic tape label checking for the RMU Backup, Backup After_Journal, Dump After_Journal, Dump Backup_File, Recover, Restore, and Restore Only_Root commands.

Table 7-17 describes the actions Oracle RMU takes when processing tape labels. The table is applicable to both OpenVMS and Digital UNIX systems except where noted.

Table 7–17 Oracle RMU Procedure to Check Tape Labels

Step	Oracle RMU Action						
❶	<p>This step applies to OpenVMS systems only. Oracle RMU checks that the first tape volume has been logically mounted.¹</p> <p>On OpenVMS systems, use the DCL MOUNT/FOREIGN command to mount the tape you want to use. You must mount the first tape volume; Oracle RMU mounts subsequent volumes automatically. If OpenVMS encounters an error when mounting the tape, Oracle RMU takes one of the following actions:</p> <ul style="list-style-type: none"> • Report the problem and retry the operation. • Abort the operation. <p>If a severe error occurs when you mount the first volume, Oracle RMU will not mount subsequent volumes. Oracle RMU tape operations do not automatically allocate the tape drives used. In an environment where many users compete for a few tape drives, it is possible for another user to seize a drive while Oracle RMU is waiting for you to load the next tape volume. You can reserve tape drives for yourself as follows:</p> <ol style="list-style-type: none"> 1. Enter the DCL ALLOCATE command for the drives you plan to use before you enter an Oracle RMU command. 2. Issue a DCL DEALLOCATE command after the Oracle RMU command completes. 						
❷	<p>Oracle RMU checks the tape characteristics.</p> <p>For an RMU Backup, RMU Backup After_Journal, or RMU Dump Backup_File command, the tape must be properly mounted and cannot be write protected. If these checks fail, Oracle RMU prompts you to mount the correct volume without write protection and indicate when you are finished.</p>						
❸	<p>Checks the tape labels² as described in the following table:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">If . . .</th> <th style="text-align: left;">Then . . .</th> </tr> </thead> <tbody> <tr> <td>The volume label disagrees with the label specified</td> <td>Oracle RMU returns an informational message.</td> </tr> <tr> <td>The tape is not the first volume, or if you specified the Rewind qualifier in the command</td> <td>Oracle RMU checks the tape protection and the expiration date of the tape and returns an informational message regarding the error. Oracle RMU checks the user-id parameter specified with the Owner=user-id qualifier of the RMU Backup or RMU Backup After_Journal command. You can write over an expired tape and you can read a tape that has not expired.</td> </tr> </tbody> </table>	If . . .	Then . . .	The volume label disagrees with the label specified	Oracle RMU returns an informational message.	The tape is not the first volume, or if you specified the Rewind qualifier in the command	Oracle RMU checks the tape protection and the expiration date of the tape and returns an informational message regarding the error. Oracle RMU checks the user-id parameter specified with the Owner=user-id qualifier of the RMU Backup or RMU Backup After_Journal command. You can write over an expired tape and you can read a tape that has not expired.
If . . .	Then . . .						
The volume label disagrees with the label specified	Oracle RMU returns an informational message.						
The tape is not the first volume, or if you specified the Rewind qualifier in the command	Oracle RMU checks the tape protection and the expiration date of the tape and returns an informational message regarding the error. Oracle RMU checks the user-id parameter specified with the Owner=user-id qualifier of the RMU Backup or RMU Backup After_Journal command. You can write over an expired tape and you can read a tape that has not expired.						

¹Not applicable to parallel backup operations.

²The label specified is either the label you specify with the Label qualifier or, if you do not specify the Label qualifier, the label Oracle RMU constructs by default from the backup file name.

(continued on next page)

Table 7–17 (Cont.) Oracle RMU Procedure to Check Tape Labels

Step	Oracle RMU Action														
4	<p>If any one of the checks fails, Oracle RMU prompts you for the action to take on the tape volume. You must select one of the following options:</p> <table border="1"> <thead> <tr> <th>Option</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Quit</td> <td> Cancels the operation, and exits from Oracle RMU.</td> </tr> <tr> <td>Override</td> <td> Ignores the failed checks and uses the tape anyway. This option is available only when the tape is performing a read request using either the RMU Dump After_Journal, RMU Dump Backup, RMU Recover, or RMU Restore command.</td> </tr> <tr> <td>Retry</td> <td> Repeats the checks after dismounting and remounting the same tape. The Retry option is useful if you left the drive off line, or if there was some other kind of correctable error.</td> </tr> <tr> <td>Unload</td> <td> Dismounts and unloads the tape so you can load the correct tape on the drive.</td> </tr> <tr> <td>Initialize</td> <td> <p>Requests that the tape be rewound and relabeled. This option is available only when you use the RMU Backup or RMU Backup After_Journal command.</p> <p>When you specify the Initialize option, Oracle RMU determines whether or not the label supplied matches the actual volume label for the tape. If the label does not match the actual volume label for the tape, Oracle RMU returns the following message:</p> <pre>This tape was incorrectly labeled. Expected AAAA03 - found AAAA55 Specify tape disposition (QUIT,INITIALIZE,RETRY,UNLOAD) RMU> Initialize</pre> <p>The error was returned because the Label qualifier AAAA03 was specified for a tape whose actual label is AAAA55. When the Initialize option is specified, the volume label of the tape is changed from AAAA55 to AAAA03.</p> </td> </tr> <tr> <td>Initialize As</td> <td> <p>Allows you to request that the tape be rewound and relabeled. The Initialize As option can be used with the RMU Backup command only. The Initialize and Initialize As options differ as follows:</p> <ul style="list-style-type: none"> • The Initialize option allows you to relabel the tape with the volume label that was originally expected. • The Initialize As option allows you to specify any volume label for the tape. <p>As with the Initialize option, you must set the label appropriately. If the RMU Backup command determines that the label name does not match the actual volume label for the tape, Oracle RMU returns the WRNGLBL message shown in the preceding Initialize option description.</p> </td> </tr> </tbody> </table>	Option	Description	Quit	Cancels the operation, and exits from Oracle RMU.	Override	Ignores the failed checks and uses the tape anyway. This option is available only when the tape is performing a read request using either the RMU Dump After_Journal, RMU Dump Backup, RMU Recover, or RMU Restore command.	Retry	Repeats the checks after dismounting and remounting the same tape. The Retry option is useful if you left the drive off line, or if there was some other kind of correctable error.	Unload	Dismounts and unloads the tape so you can load the correct tape on the drive.	Initialize	<p>Requests that the tape be rewound and relabeled. This option is available only when you use the RMU Backup or RMU Backup After_Journal command.</p> <p>When you specify the Initialize option, Oracle RMU determines whether or not the label supplied matches the actual volume label for the tape. If the label does not match the actual volume label for the tape, Oracle RMU returns the following message:</p> <pre>This tape was incorrectly labeled. Expected AAAA03 - found AAAA55 Specify tape disposition (QUIT,INITIALIZE,RETRY,UNLOAD) RMU> Initialize</pre> <p>The error was returned because the Label qualifier AAAA03 was specified for a tape whose actual label is AAAA55. When the Initialize option is specified, the volume label of the tape is changed from AAAA55 to AAAA03.</p>	Initialize As	<p>Allows you to request that the tape be rewound and relabeled. The Initialize As option can be used with the RMU Backup command only. The Initialize and Initialize As options differ as follows:</p> <ul style="list-style-type: none"> • The Initialize option allows you to relabel the tape with the volume label that was originally expected. • The Initialize As option allows you to specify any volume label for the tape. <p>As with the Initialize option, you must set the label appropriately. If the RMU Backup command determines that the label name does not match the actual volume label for the tape, Oracle RMU returns the WRNGLBL message shown in the preceding Initialize option description.</p>
Option	Description														
Quit	Cancels the operation, and exits from Oracle RMU.														
Override	Ignores the failed checks and uses the tape anyway. This option is available only when the tape is performing a read request using either the RMU Dump After_Journal, RMU Dump Backup, RMU Recover, or RMU Restore command.														
Retry	Repeats the checks after dismounting and remounting the same tape. The Retry option is useful if you left the drive off line, or if there was some other kind of correctable error.														
Unload	Dismounts and unloads the tape so you can load the correct tape on the drive.														
Initialize	<p>Requests that the tape be rewound and relabeled. This option is available only when you use the RMU Backup or RMU Backup After_Journal command.</p> <p>When you specify the Initialize option, Oracle RMU determines whether or not the label supplied matches the actual volume label for the tape. If the label does not match the actual volume label for the tape, Oracle RMU returns the following message:</p> <pre>This tape was incorrectly labeled. Expected AAAA03 - found AAAA55 Specify tape disposition (QUIT,INITIALIZE,RETRY,UNLOAD) RMU> Initialize</pre> <p>The error was returned because the Label qualifier AAAA03 was specified for a tape whose actual label is AAAA55. When the Initialize option is specified, the volume label of the tape is changed from AAAA55 to AAAA03.</p>														
Initialize As	<p>Allows you to request that the tape be rewound and relabeled. The Initialize As option can be used with the RMU Backup command only. The Initialize and Initialize As options differ as follows:</p> <ul style="list-style-type: none"> • The Initialize option allows you to relabel the tape with the volume label that was originally expected. • The Initialize As option allows you to specify any volume label for the tape. <p>As with the Initialize option, you must set the label appropriately. If the RMU Backup command determines that the label name does not match the actual volume label for the tape, Oracle RMU returns the WRNGLBL message shown in the preceding Initialize option description.</p>														
5	<p>Remounts tapes and re verifies tape labels, as requested.</p> <p>Changing one tape for another tape follows essentially the same procedure, with one difference. If the first volume is not at the beginning-of-tape (BOT) mark, complete label checking is not available.</p>														

7.13.5 Monitoring Error Rates

You should monitor the error rates on the tape devices you use for backup operations. Although Oracle RMU ignores most tape errors, all errors detected by the hardware might not be reported to Oracle RMU and all errors cannot be recovered reliably.

If you observe errors during a backup operation, you should take precautions against possible effects on the backup file. For example, you can use the RMU Restore or RMU Dump Backup commands to verify the backup file is readable or you can perform another backup operation to a different device with different tapes to create an alternative backup file. Large numbers of errors indicate the need for maintenance on your tape drive, or poor media quality. If you have bad media, you should take corrective action as soon as possible.

The following list contains operating-system specific recommendations:

Digital UNIX

- On Digital UNIX systems, if you enter an RMU command interactively and a tape request or problem arises, Oracle RMU notifies the person who issued the command. After being notified of the problem, the user who issued the command can either fix the problem (if the user has access to the tape drive) or contact the tape operator for assistance. ♦

OpenVMS OpenVMS
VAX Alpha

- On OpenVMS systems, if you enter an Oracle RMU command from an OpenVMS batch job, Oracle RMU reports tape requests and problems to the tape operator. This occurs because tape requests and problems often require manual intervention, and if the RMU command was issued from a batch job, the only person who might be available is the operator.

On OpenVMS systems, if you enter an Oracle RMU command interactively and a tape request or problem arises, Oracle RMU notifies the person who issued the command through the I/O channel assigned to the logical name SYS\$COMMAND.

You can use the DCL command REQUEST to notify the tape operator as shown in the following example:

```
$ REQUEST/REPLY/TO=TAPES -  
_$_ "Please Write Enable tape ATOZBG on drive $255$MUA6:"
```

♦

7.14 Displaying Database Backup Information

When you perform a full backup operation, the database root file is updated with information such as the type of backup operation (full, incremental, or by-area), what areas are backed up, the name of the database, when the database backup operation occurred, the location of the database root file, the process name and user identifier for the user who performed the operation, and so forth.

Example 7–14 shows sample output from an RMU Dump Backup_File command.

Example 7–14 Sample Oracle RMU Dump Backup_File Commands

```
$ RMU/DUMP/BACKUP_FILE/OPTIONS=(DATA,FULL)/PROCESS=RECORD=1 MFPERS.RBF
.
.
Oracle Rdb specific root record

DBKEY for Oracle Rdb bootstrap page is 8:440:0
Release retrieval locks when no longer needed
Do not wait on record lock conflicts
Latest full backup file is dated 5-NOV-1995 15:29:37.23
Latest full backup transaction sequence number is 76
Database has never been incrementally restored
Database has never been fully restored
Latest full verify occurred at 5-NOV-1995 14:07:27.72
Database has never been altered
.
.
Snapshot area for storage area RESUMES
.
.
HEADER_SIZE = 80          OS_ID = 1024    UTILITY_ID = 722
APPLICATION_TYPE = 1 SEQUENCE_NUMBER = 1 MAJ_VER = 1 MIN_VER = 1
VOL_NUMBER = 1 BLOCK_SIZE = 32256      CRC = A2C8B8E1 NOCRC = 00
CRC_ALTERNATE = 00      BACKUP_NAME = MFPERS.RBF      AREA_ID = 1
HIGH_PNO = 152 LOW_PNO = 1 HDR_CHECKSUM = EEA8

REC_SIZE = 287 REC_TYPE = 1 BADDATA = 00 ROOT = 01 AREA_ID = 0
LAREA_ID = 0 PNO = 0
```

(continued on next page)

Example 7–14 (Cont.) Sample Oracle RMU Dump Backup_File Commands

```
Backup File Summary Record, 287. bytes
BACKUP_NAME = RMU/BACKUP      USERNAME = ORION    UIC = [999211,000001]
Root creation date = 4-NOV-1995 14:34:17.20
Backup date = 5-NOV-1995 15:29:37.23
Last Full Backup date = 17-NOV-1858 00:00:00.00 OS type = 1024
OS Version = V5.4             System ID register = 301989891
Utility = Oracle Rdb Management Utility V7.0      Major version = 70
Minor version = 0   BLOCK_SIZE = 32256   GROUP_SIZE = 0   ACTIVE_IO = 3
FULL_BACKUP = 1   ONLINE_BACKUP = 0      AREAS_EXCLUDED = 0
USED_PAGE_MODIFIED_MAP = 00      VOLUME_SET =
VOLUME_NUMBER = 1      LIVE_AREAS = 10   VOLUME_AREAS = 10

BACKUP FILE = MFPERS.RBF
COMMAND = RMU/BACKUP MF_PERSONNEL MFPERS.RBF
ROOT_FILE = DUA01:[TEST1]MF_PERSONNEL.RDB;1
DRIVE = _$111$DUA99

Volume area # 1 AREA DBID = 0   STARTING PNO = 0
Volume area # 2 AREA DBID = 0   STARTING PNO = 0
Volume area # 3 AREA DBID = 0   STARTING PNO = 0
Volume area # 4 AREA DBID = 0   STARTING PNO = 0
Volume area # 5 AREA DBID = 0   STARTING PNO = 0
Volume area # 6 AREA DBID = 0   STARTING PNO = 0
Volume area # 7 AREA DBID = 0   STARTING PNO = 0
Volume area # 8 AREA DBID = 0   STARTING PNO = 0
Volume area # 9 AREA DBID = 0   STARTING PNO = 0
```

The `RMU Dump Backup_File` command provides a number of qualifiers and options to help you customize the level of detail in the output display. Example 7–14 uses the `Options=(Data,Full)` to include a dump of the database backup file header information and the contents of the backup file's records and blocks.

You can use the Oracle `RMU Dump` command to capture particular information, such as the date of the last backup file. For example, you can dump the date in the database root file and compare it with the date in the backup file to confirm that this is your last backup file.

To display this information, use the `RMU Dump Backup_File` command in its default mode with no qualifiers. Oracle `RMU` returns the following message during the integrity check of the database backup file:

```
RMU-I-DMPTXT_163, No dump output selected. Performing read check.
```

If you want to display the contents of the database root file as recorded in the backup file and also check the integrity of the backup file, specify the Options=Root qualifier, as shown in Example 7-15.

Example 7-15 Checking the Backup File to See When It Was Created

```
$ RMU/DUMP/BACKUP_FILE /OPTIONS=ROOT $111$MUA0:PERS_FULL.RBF
Database Parameters:
Root filename is "DUA01:[ORION]MF_PERSONNEL.RDB;1"
Created at 4-NOV-1995 14:34:17.20
Oracle Rdb structure level is 70.0
.
.
Database root file ACL
(IDENTIFIER=[RDB,WARD],ACCESS=READ+WRITE+CONTROL+RMU$ALTER+RMU$ANALYZE+
RMU$BACKUP+RMU$CONVERT+RMU$COPY+RMU$DUMP+RMU$LOAD+RMU$MOVE+RMU$OPEN+
RMU$RESTORE+RMU$SECURITY+RMU$SHOW+RMU$UNLOAD+RMU$VERIFY)
DBKEY for Oracle Rdb bootstrap page is 8:440:0
Release retrieval locks when no longer needed
Do not wait on record lock conflicts
Latest full backup file is dated 5-NOV-1995 15:29:37.23 <-----Notice
Latest full backup transaction sequence number is 76
Database has never been incrementally restored
Database has never been fully restored
Latest full verify occurred at 5-NOV-1995 14:07:27.72
Database has never been altered
.
.
.
```

Restoring Your Database

If you lose access to your database because of a hardware or software failure, you can use the RMU Restore command to restore a complete copy of a database backup (.rbf) file created by the RMU Backup command.

In some cases, the RMU Restore command also can restore a corrupt database. However, restoring data after possible corruption requires that you take extra precautions to make sure the backup file is not corrupt.

The RMU Restore command provides a wide variety of options for restoring a database. This chapter provides an overview of the RMU restore operation and describes many of the qualifiers that you can use with the RMU Restore command. Refer to the *Oracle RMU Reference Manual* for the complete list of RMU Restore command options.

Note

You cannot restore a database remotely across a DECnet network. This is a DECnet restriction. The DECnet software does not support disk ancillary control process (ACP) queue I/O (QIO) request system service access over the network through the file access listener (FAL).

8.1 Preparing to Restore a Database

The following list describes some tasks you should perform as a part of the database restore operation:

- Be sure that you have enough disk space to hold the new database. The RMU Restore command aborts if there is not enough disk space, resulting in a corrupt database. Should this happen at your site, you must create enough free space on the disk and run the RMU Restore command again.

- Make sure you have the last full backup file and the last incremental backup file or files (if any). The backup dates and times are recorded in the database root (.rdb) file header. The restore procedure is the same for all backup files if they were created on line or off line.
- If after-image journaling is enabled, use the RMU Recover command after restoring the database to roll your database forward to the most recently completed database commit operation. RMU Restore does this automatically, if possible.
- Ensure that sufficient Oracle RMU privileges are granted to the database creator and other users, especially if you restore the database into a directory that is owned by a resource identifier. Section 8.2 describes how to check for and grant Oracle RMU privileges. ♦

Note

If you restore (rather than create) a database to its original location, do not delete the original version of the database root file before restoring the database. When restoring the database, use the `New_Version` qualifier on the RMU Restore command. This keeps the same ACL for the new root file that exists on the original root file. After the Oracle RMU restore operation, you can delete the original root file.

8.2 Access Privileges for a Restored Database

Database access is determined by the access control entry (ACE) privileges for the OpenVMS directory into which you restore the database.

The following table describes the conditions that determine database access after an Oracle RMU restore operation:

When . . .	Then . . .
You create, restore, or copy a database into your own directory	You are the owner of both the directory and the database. Therefore, you automatically are granted all Oracle RMU privileges to access that database.
You create, restore, or copy a database into a directory owned by a resource identifier	The resource identifier is the owner of the database. The default OpenVMS privileges for the directory (those of the resource identifier) take precedence over the Oracle RMU privileges. Thus, even though you created or restored the database, you cannot access the database.

When you restore a database, ACE privileges for the database root file are added to the OpenVMS access control list (ACL). The default OpenVMS ACE privileges always appear in the ACL before the Oracle RMU ACEs (if any). When you enter an RMU command that requires access to the database, OpenVMS searches entries in the ACL in the following order:

1. The OpenVMS directory ACE

The OpenVMS ACL controls users' access to files and directories on the system. At the OpenVMS level, you can grant READ, WRITE, EXECUTE, DELETE, CONTROL, or NONE (no privileges) on a file or directory.

2. The Oracle RMU ACE

In addition to the OpenVMS ACL, Oracle RMU provides another level of security that controls the RMU functions that a user can perform. Oracle RMU manages this additional level of security by recording Oracle RMU ACE privileges in the OpenVMS ACL.

The OpenVMS process searches each entry in the ACL looking for an ACE that matches your user identification code (UIC) or the resource identifier (whichever appears first). If the ACL for a directory does not include Oracle RMU privileges, you cannot perform any Oracle RMU functions on the database.

For example, the following DCL command displays OpenVMS and Oracle RMU privileges for the TEST_DATA database:

```
$ SET DEF RDBVMS_USER1:[ORAUSER.MYDATABASE]
$ DIRECTORY /SECURITY
TEST_DATA.RDB;1 14-FEB-1996 14:07:40.81 [RDB,ORAUSER] (RWED,RW,,)
  (IDENTIFIER=[RDB,ORAUSER], ❶ACCESS=READ+WRITE+CONTROL+❷BIT_5+BIT_6+
  BIT_7+BIT_8+BIT_9+BIT_10+BIT_11+BIT_12+BIT_13+BIT_14+BIT_15+BIT_16+
  BIT_17+BIT_18)
TEST_DATA.SNP;1 14-FEB-1996 14:07:42.14 [RDB,ORAUSER] (RWED,RW,,)
```

❶ OpenVMS access privileges

❷ Oracle RMU privileges (BIT_5 through BIT_18)

Because the OpenVMS operating system is unable to translate Oracle RMU access privileges, the DCL command displays them in a BIT_n format.

You can use the RMU Show Privileges command to translate the BIT_n names, as shown in the following example:

```

$ SET DEF RDBVMS_USER1:[ORAUER.MYDATABASE]
$ RMU/SHOW PRIVILEGE TEST_DATA.RDB
Object type: file, Object name:RDBVMS_USER1:[ORAUER.MYDATABASE]TEST_DATA.RDB;1,
on 14-FEB-1996 14:11:19.96
  (IDENTIFIER=[RDB,ORAUER], ❶ACCESS=READ+WRITE+CONTROL+❷RMU$ALTER+
  RMU$ANALYZE+RMU$BACKUP+RMU$CONVERT+RMU$COPY+RMU$DUMP+RMU$LOAD+
  RMU$MOVE+RMU$OPEN+RMU$RESTORE+RMU$SECURITY+RMU$SHOW+RMU$UNLOAD+
  RMU$VERIFY)

```

❶ OpenVMS access privileges

❷ Oracle RMU privileges

The Oracle RMU Show Privilege command displays the Oracle RMU access privileges as *RMU\$Function*, where *Function* is alter, analyze, and so forth. In the example, all of the Oracle RMU privileges are listed. Thus, the database user ORAUER can perform all Oracle RMU operations on the database.

Note

Access to an Oracle Rdb database is controlled by ACLs that are stored in the database. Because Oracle Rdb is privileged, it can always open the database to check these privileges regardless of OpenVMS privileges.

If you create, copy, or restore a database to a directory owned by a resource identifier, the ACE for the database inherits the default ACE defined for the directory. For example:

```

$ SET FILE/ACL=(ID=DATABASE_1,ACCESS=READ+WRITE+EXECUTE+DELETE+CONTROL) -
_$ DB_DIRECTORY.DIR
$ SET FILE/ACL=(ID=DATABASE_1, OPTIONS=DEFAULT, -
_$ ACCESS=READ+WRITE+EXECUTE+DELETE+CONTROL) DB_DIRECTORY.DIR
$ SET FILE DB_DIRECTORY.DIR/OWNER=DATABASE_1
$ DIRECTORY/SECURITY DB_DIRECTORY.DIR

Directory RDBVMS_USER1:[ORAUER]

DB_DIRECTORY.DIR;1 14-FEB-1996 16:55:08.86  DATABASE_1      (RWE,RWE,RE,E)
  (IDENTIFIER=DATABASE_1,OPTIONS=DEFAULT,ACCESS=READ+WRITE+EXECUTE+
  DELETE+CONTROL)
  (IDENTIFIER=DATABASE_1,ACCESS=READ+WRITE+EXECUTE+DELETE+CONTROL)
. . .
$ RMU /RESTORE TEST_DATABASE.RBF
$ DIRECTORY/SECURITY

Directory RDBVMS_USER1:[ORAUER.DB_DIRECTORY]

```

```

TEST_DATABASE.RDB;1 14-FEB-1996 16:57:18.20 DATABASE_1 (RWED,RW,,)
  (IDENTIFIER=[RDB,ART],OPTIONS=NOPROPAGATE,ACCESS=READ+WRITE+
CONTROL)
  (IDENTIFIER=DATABASE_1,ACCESS=READ+WRITE+EXECUTE+DELETE+CONTROL)

```

Total of 1 file.

```

TEST_DATABASE.SNP;1 14-FEB-1996 16:57:19.16 DATABASE_1 (RWED,RW,,)
  (IDENTIFIER=[RDB,ART],OPTIONS=NOPROPAGATE,ACCESS=READ+WRITE+
CONTROL)
  (IDENTIFIER=DATABASE_1,ACCESS=READ+WRITE+EXECUTE+DELETE+CONTROL)

```

The example ACE for the user ORAUSER shows that the default OpenVMS privileges are applied when you create a file in a directory owned by a resource identifier. That is, the file system supplies an ACE that grants the user CONTROL access plus the access specified in the Owner field of the UIC-based protection. Thus, user [RDB,ORAUSER] inherits READ+WRITE from the database UIC of (RWED,RW,,), plus the CONTROL privilege that the file system grants by default.

OpenVMS has removed the RMU entries from the ACL. Therefore, any attempts to perform RMU operations to the restored database are denied. You can resolve the situation using either of the following methods:

- From an OpenVMS account with BYPASS privileges, use the RMU Set Privilege command to grant the necessary Oracle RMU privileges to each user of the database.
- Use the DCL command SET FILE/ACL to set up the database root file ACE in the directory's default ACL, and set up the Oracle RMU access privileges. Using the SET FILE/ACL command applies the same ACL to all files including non-database files. Thus, ACLs for non-database files inherit the Oracle RMU privileges you set up in the directory's default ACL for the database root file. (In this case, you cannot use the RMU Set Privilege command; it fails because you do not have the Oracle RMU privileges to change RMU security.)

◆

8.3 Full Database Restore Operations

Use your full database backup file to restore your database to the state it was in when the last full backup was made. If you are unsure which database backup file (.rbf) file is the most current, use the RMU Dump Backup_File Options=Root command to display the date of the last full backup operation (see Section 7.7).

The full backup file should have the same timestamp as the one shown in the database root (.rdb) file header. When you find the correct database .rbf file, use the RMU Restore command to restore your database. For example, to restore from the DBS_BACKUPS disk, issue the command shown in Example 8–1.

Example 8–1 Starting a Full Restore Operation from the DBS_BACKUPS Disk

```
$ RMU/RESTORE/LOG DBS_BACKUPS:[MFPERS]MF_PERS_FULL.RBF
%RMU-I-RESTXT_04, Thread 1 file uses devices DBM$DISK:
%RMU-I-AIJRSTBEG, restoring after-image journal "state" information
%RMU-I-AIJRSTJRN, restoring journal "JOURN_1" information
%RMU-I-AIJRSTSEQ, journal sequence number is "0"
%RMU-I-AIJRSTSUC, journal "JOURN_1" successfully restored from file
"DISK11$:[JOURNAL]JOURN_1.AIJ;1"
%RMU-I-AIJRSTJRN, restoring journal "JOURN_2" information
%RMU-I-AIJRSTSEQ, journal sequence number is "1"
%RMU-I-AIJRSTSUC, journal "JOURN_2" successfully restored from file
"DISK12$:[JOURNAL]JOURN_2.AIJ;1"
%RMU-I-AIJRSTJRN, restoring journal "JOURN_3" information
%RMU-I-AIJRSTSEQ, journal sequence number is "2"
%RMU-I-AIJRSTSUC, journal "JOURN_3" successfully restored from file
"DISK13$:[JOURNAL]JOURN_3.AIJ;1"
%RMU-I-AIJRSTEND, after-image journal "state" restoration complete
%RMU-I-RESTXT_00, restored root file DB_DISK:[MFPERS]MF_PERSONNEL.RDB;1
%RMU-I-LOGRESSST, restored storage area DISK1:[MFPERS]MF_PERS_DEFAULT.RDA;1
%RMU-I-LOGRESSST, restored storage area DISK3:[MFPERS]EMPIDS_LOW.RDA;1
%RMU-I-LOGRESSST, restored storage area DISK5:[MFPERS]EMPIDS_MID.RDA;1
%RMU-I-LOGRESSST, restored storage area DISK6:[MFPERS]EMPIDS_OVER.RDA;1
%RMU-I-LOGRESSST, restored storage area DISK3:[MFPERS]DEPARTMENTS.RDA;1
%RMU-I-LOGRESSST, restored storage area DISK5:[MFPERS]SALARY_HISTORY.RDA;1
%RMU-I-LOGRESSST, restored storage area DISK3:[MFPERS]JOBS.RDA;1
%RMU-I-LOGRESSST, restored storage area DISK4:[MFPERS]EMP_INFO.RDA;1
%RMU-I-LOGRESSST, restored storage area DISK2:[MFPERS]RESUME_LISTS.RDA;2
%RMU-I-LOGRESSST, restored storage area DISK6:[MFPERS]RESUMES.RDA;2
%RMU-I-LOGRESSST, restored storage area DISK3:[MFPERS]EMPIDS_LOW.RDA;2
%RMU-I-RESTXT_05, rebuilt 1 space management page
%RMU-I-RESTXT_06, restored 0 inventory pages
%RMU-I-RESTXT_07, rebuilt 0 logical area bitmap pages
%RMU-I-RESTXT_08, restored 51 data pages
%RMU-I-LOGRESSST, restored storage area DISK5:[MFPERS]EMPIDS_MID.RDA;2
%RMU-I-RESTXT_05, rebuilt 1 space management page
%RMU-I-RESTXT_06, restored 0 inventory pages
%RMU-I-RESTXT_07, rebuilt 0 logical area bitmap pages
%RMU-I-RESTXT_08, restored 51 data pages
```

(continued on next page)

Example 8-1 (Cont.) Starting a Full Restore Operation from the DBS_BACKUPS Disk

```
.
.
%RMU-I-LOGRESSST, restored storage area DISK1:[MFPERS]MF_PERS_DEFAULT.RDA;2
%RMU-I-RESTXT_05, rebuilt 1 space management page
%RMU-I-RESTXT_06, restored 6 inventory pages
%RMU-I-RESTXT_07, rebuilt 135 logical area bitmap pages
%RMU-I-RESTXT_08, restored 576 data pages
%RMU-I-RESTXT_01, Initialized snapshot file DISK1:[MFPERS]MF_PERS_DEFAULT.SNP;2
%RMU-I-LOGINIFIL, contains 248 pages, each page is 2 blocks long
%RMU-I-RESTXT_01, Initialized snapshot file DISK4:[MFPERS]EMPIDS_LOW.SNP;2
%RMU-I-LOGINIFIL, contains 10 pages, each page is 2 blocks long
.
.
%RMU-I-AIJWASON, AIJ journaling was active when the database was backed up
%RMU-I-AIJREFUL, Recovery of the entire database starts with AIJ file
sequence 0
%RMU-I-AIJRECBEG, recovering after-image journal "state" information
%RMU-I-AIJRSTAVL, 3 after-image journals available for use
%RMU-I-AIJRSTMOD, 3 after-image journals marked as "modified"
%RMU-F-AIJENBOVR, enabling AIJ journaling would overwrite an existing journal
%RMU-I-AIJISOFF, after-image journaling has been disabled
%RMU-I-AIJRECEND, after-image journal "state" recovery complete
%RMU-I-LOGRECEDB, recovering database file DB_DISK:[MFPERS]MF_PERSONNEL.RDB;1
%RMU-I-AIJAUTOREC, starting automatic after-image journal recovery
%RMU-I-LOGOPNAIJ, opened journal file DISK11$:[JOURNAL]JOURN_1.AIJ;1
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-LOGRECOVR, 2 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 1 transaction ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCEC, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence number
needed will be 1
%RMU-I-LOGOPNAIJ, opened journal file DISK12$:[JOURNAL]JOURN_2.AIJ;1
%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed
%RMU-I-LOGRECOVR, 2 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCEC, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence
number needed will be 2
%RMU-I-LOGOPNAIJ, opened journal file DISK13$:[JOURNAL]JOURN_3.AIJ;1
%RMU-I-AIJONEDONE, AIJ file sequence 2 roll-forward operations completed
%RMU-I-LOGRECOVR, 0 transactions committed
```

(continued on next page)

Example 8–1 (Cont.) Starting a Full Restore Operation from the DBS_BACKUPS Disk

```
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCE, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence
number needed will be 3
%RMU-I-AIJALDONE, after-image journal roll-forward operations completed
%RMU-I-LOGSUMMARY, total 4 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
%RMU-I-LOGSUMMARY, total 1 transaction ignored
%RMU-I-AIJSUCCE, database recovery completed successfully
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence
number needed will be 3
%RMU-I-AIJNOENABLED, after-image journaling has not yet been enabled
%RMU-I-LOGINTEGRATE, Integrating DB_DISK:[MFPERS]MF_PERSONNEL.RDB;1 into CDD path
MF_PERSONNEL
$
```

A restore operation automatically applies the journal files if they are available, modified, and on disk. If you restore a database for which journals exist but are unavailable because they are backed up, or they are located on a disk that is off line, Oracle RMU returns the following warning message indicating that you must recover the journals manually:

```
%RMU-W-USERECCOM, Use the RMU Recover command. The journals are not available.
```

If you want to inhibit automatic recovery because you want to restore incrementally, you must explicitly prevent recovery by specifying:

- The Norecovery qualifier on the RMU Restore command line
- A new journal state with one of the journal options

In Example 8–1, no previous version of the database exists. If a previous version exists and you omit the New_Version qualifier, Oracle RMU returns the error message shown in Example 8–2.

Example 8–2 Omitting the New_Version Qualifier in a Restore Operation When a Previous Version of the Database Exists

```
$ RMU/RESTORE/LOG DBS_BACKUPS:[MFPERS]MF_PERS_FULL.RBF
%RMU-I-REXTXT_04, Thread 1 file uses devices DBM$DISK:
%RMU-F-FILACCERR, error creating database file DB_DISK:[MFPERS]MF_PERSONNEL.RDB;1
-RMS-E-FEX, file already exists, not superseded
```

Use the `New_Version` qualifier to create a new version of the database as shown in Example 8–3.

Example 8–3 Using the New_Version Qualifier During a Restore Operation to Supersede a Previous Version of the Database

```
$ RMU/RESTORE /NEW_VERSION /LOG DBS_BACKUPS:[MFPERS]MF_PERS_FULL.RBF
%RMU-I-REXTXT_04, Thread 1 file uses devices DBM$DISK:
.
.
.
```

When the restore operation proceeds using the `New_Version` qualifier, new version numbers for the `.rdb`, `.rda`, and `.snp` files are created. You must specify the new version number of the `.rdb` file if you want to apply an incremental restore operation, or roll the database forward from the `.ajj` file, or both. Otherwise, if you perform an `RMU Verify Root` command, Oracle `RMU` returns an error message indicating that the `.ajj` file contains references to the older database file. See Example 9–7 and Example 9–15 for examples of rollforward operations that use the `RMU Recover` command. If any old versions of your database exist when you are finished with the full incremental restore and rollforward operations, delete them as shown in Example 8–4.

Example 8–4 Deleting an Old Database Version After Fully Restoring and Recovering the Database

```
$ SQL
SQL> DROP DATABASE FILENAME DB_DISK:[MFPERS]MF_PERSONNEL.RDB;1;
```

8.4 Incremental Restore Operations

After you restore your database from a full .rbf file, you can also restore your database from an incremental .rbf file, if one exists. If you intend to restore an incremental backup file, you should use the Norecovery qualifier on all but the last restore operation to be performed. The Norecovery qualifier inhibits the automatic recovery from journal files.

Always use the last incremental .rbf file taken since the last full backup operation; make sure the timestamp of the incremental .rbf file is later than the timestamp of the last full backup file. Use the RMU Dump Backup_File Options=Root command to determine from the .rdb file if the database was ever incrementally restored, as shown in Example 8-5.

Example 8-5 Displaying the Root File Header to Check If the Database Was Ever Incrementally Restored

```
$ RMU/DUMP/BACKUP_FILE /OPTIONS=ROOT -
_ $ DBS_BACKUPS:[MFPERS]MF_PERS_FULL.RBF
Database parameters
.
.
.
Latest full backup file is dated 20-OCT-1993 14:03:06.47
Database has never been incrementally restored
```

This mf_personnel database has never been incrementally restored. Use the RMU Restore Incremental command to restore your database incrementally as shown in Example 8-6.

Example 8-6 Starting an Incremental Restore Operation Following a Full Restore Operation

```
$ RMU/RESTORE/NOLOG/NEW_VERSION/NORECOVERY -
_ $ DBS_BACKUPS:[MFPERS]MF_PERS_FULL.RBF
$ RMU/RESTORE/INCREMENTAL/ROOT=DB_DISK:[MFPERS]MF_PERSONNEL.RDB;3
Backup: DBS_BACKUPS:[MFPERS]MF_PERS_INCR.RBF /LOG

%RMU-I-RESTXT_04, Thread 1 file uses devices DBM$DISK:
DB_DISK:[MFPERS]MF_PERSONNEL.RDB;3, restore incrementally? [N]:Y
```

(continued on next page)

Example 8–6 (Cont.) Starting an Incremental Restore Operation Following a Full Restore Operation

```
%RMU-I-RESTXT_04, Thread 1 file uses devices DBM$DISK:
.
.
.
%RMU-I-LOGINTEGRATE, Integrating DB_DISK:[MFPERS]MF_PERSONNEL.RDB;3 into path
MF_PERSONNEL
```

Note

Automatic .ajj file recovery will be attempted when you restore a database from a full, incremental, by-area, or by-page backup file. Use the Norecovery qualifier to disable this feature. For example, if you intend to restore a database as in this case by first issuing a full RMU Restore command followed by the application of one or more RMU Restore Incremental or RMU Restore Area commands, you must specify the Norecovery command on all but the last RMU Restore command in the series you intend to issue. Allowing Oracle RMU to attempt automatic recovery with a full restore operation when you intend to apply additional incremental, by-area, or by-page backup files may result in a corrupt database.

The Root qualifier in Example 8–6 specifies the correct version of the database to restore incrementally (version 3 of the .rdb file). Use the RMU Dump Backup_File Options=Root command to verify the effect of the incremental restore operation as shown in Example 8–7.

Example 8–7 Displaying the Root File Header to Check if the Database Was Restored Incrementally

```
$ RMU/DUMP/BACKUP_FILE /OPTIONS=ROOT MF_PERS_INCR.RBF
.
.
.
Latest full backup file is dated 20-OCT-1993 14:03:06.47
Latest full backup transaction sequence number is 1
Latest incremental restore file used is dated 1-NOV-1993 13:03:31:78
```

The .rdb file header now shows the timestamp of the incremental restore file. You cannot now use the same incremental .rbf file again for this database. If you try to restore your database incrementally by using the same incremental .rbf file a second time, you receive the message shown in Example 8–8.

Example 8–8 Error If You Try to Incrementally Restore the Same Database a Second Time, Using the Same Incremental .rbf File

```
$ RMU/RESTORE/INCREMENTAL/ROOT=DB_DISK:[MFPERS]MF_PERSONNEL.RDB;3/LOG
Backup: DBS_BACKUPS:[MFPERS]MF_PERS_INCR.RBF

%RMU-I-REXTXT_04, Thread 1 file uses devices DB_DISK:
DB_DISK:[MFPERS]MF_PERSONNEL.RDB;3, restore incrementally? [N]:Y
%RMU-F-INCAPPLIED, incremental restore to 1-NOV-1993 13:03:31:78 has already been done
```

When restoring a database to the same device and directory as an existing copy of the same database, you must use the `New_Version` qualifier in the `RMU Restore` command and the `Root` qualifier in the `RMU Restore Incremental` command. Otherwise, you may apply your incremental .rbf file against the wrong version of the database.

The confirmation prompt of the `RMU Restore Incremental` command asks you to make sure you are applying your incremental .rbf file to the right version of the right database. If you are not certain, press the Return key to accept the default response of N (no) and cancel the incremental restore operation.

Caution

Make sure no one else accesses your database between issuing an `RMU Restore` command and issuing an `RMU Restore Incremental` command. That is, close the database so that it is inaccessible. Updates made between a full and an incremental restore operation can be lost if the incremental restore operation writes over the database page on which such an update is made.

8.5 A Sample Restore Procedure

This section describes a sample restore procedure as part of a possible strategy for restoring databases from regularly created and maintained database backup files.

Assume you lose access to your database because the disk on which the database resides has a read/write head failure and the .rdb file is damaged. After the disk is repaired, you want to restore your database to its original location and delete the damaged database.

Assume the disk head failure occurs on a Wednesday. First, you must restore the full .rbf file made the previous Friday and the incremental database .rbf file made Tuesday night, the day before the disk failure. After you mount the appropriate tape volumes, you can issue the commands shown in Example 8–9 to restore your database from tape and delete the damaged version. If the .rdb file is damaged, you must use the operating system command to delete the file.

Example 8–9 Sample Restore Procedure Followed by Deleting an Old Version of the Database

```
$ RMU/RESTORE /NORECOVERY $111$MUA0:PERS_FULL.RBF
$ RMU/RESTORE/INCREMENTAL/ROOT=DB_DISK:[MFPERS]MF_PERSONNEL.RDB;
Backup: DBS_BACKUPS:PERS_INCR.RBF /LOG

%RMU-I-RESTXT_04, Thread 1 file uses devices DBM$DISK:
DB_DISK:MF_PERSONNEL.RDB;3, restore incrementally? [N]:Y
$
$ DELETE DB_DISK:[MFPERS]MF_PERSONNEL.RDB;2
```

Watch the database version numbers carefully to make sure you restore your incremental backup file against the right database. In this example, a semicolon (;) at the end of the .rdb file name indicates the incremental restore procedure is to be applied to the highest version of the .rdb file. This is only a problem if you use the same directory for two copies of the database. If the original database is still present, the restore operation will not retain .aij files and automatic recovery will not be performed after the incremental restore operation.

The last step following a full and incremental restore operation is to roll the database forward by using the RMU Recover command. This brings your database up-to-date with the contents of the .aij file. See Chapter 9 for an example and the steps to follow to recover or roll a database forward.

8.6 Performing By-Area Restore Operations

The RMU Restore command permits you to restore one or more storage areas on line (using the Online qualifier) without restoring the entire database. Being able to restore one or more storage areas on line simplifies the physical restructuring of a large database. However, it is strongly recommended that you enable after-image file journaling to back up and restore databases by storage area and to roll the database forward to the most recently completed transaction in the event of a problem. Users can be attached to the database but are excluded from accessing the area being restored because the online by-area restore operation takes out an exclusive lock on the area being restored.

See the note in Section 8.4 about using the Norecovery qualifier if more than one area is to be restored in successive Oracle RMU commands so you can disable recovery on all but the last RMU Restore command.

Using the Area and Online qualifiers, you can create a storage area selection list of only the storage areas you want restored on line. By default, Area is not specified, and all storage areas in the backup file are restored off line. For example, suppose you wanted to restore the EMP_INFO storage area on line and automatically roll forward on line the contents of the .aij file for this storage area up to the current date. Example 8–10 shows this scenario.

Example 8–10 Restoring and Recovering the EMP_INFO Storage Area

```
$ ! Restore on line only the EMP_INFO storage area.
$
RMU/RESTORE/NOCCDD_INTEGRATE/AREA/LOG/ONLINE MFPERS.RBF EMP_INFO
%RMU-I-RESTXT_04, Thread 1 uses devices DUA1:
%RMU-I-LOGRESSST, restored storage area DUA1:[ORION]EMP_INFO.RDA;1
%RMU-I-LOGRESSST, restored storage area DUA1:[ORION]EMP_INFO.RDA;1
%RMU-I-RESTXT_05, rebuilt 1 space management page
%RMU-I-RESTXT_06, restored 0 inventory pages
%RMU-I-RESTXT_07, rebuilt 0 logical area bitmap pages
%RMU-I-RESTXT_08, restored 30 data pages
%RMU-I-RESTXT_01, Initialized snapshot file DUA2:[ORION]EMP_INFO.SNP;1
%RMU-I-LOGINIFIL, contains 10 pages, each page is 2 blocks long
%RMU-I-AIJWASON, AIJ journaling was active when the database was backed up
%RMU-I-AIJRECFUL, Recovery of the entire database starts with AIJ file
sequence 0
%RMU-I-AIJRECARE, Recovery of area EMP_INFO starts with AIJ file sequence 0
%RMU-I-AIJBADAREA, inconsistent storage area DUA1:[ORION]EMP_INFO.RDA;1
needs AIJ sequence number 0
%RMU-I-LOGRECDB, recovering database file DUA3:[ORION]MF_PERSONNEL.RDB;1
%RMU-I-AIJAUTOREC, starting automatic after-image journal recovery
%RMU-I-LOGOPNAIJ, opened journal file DUA11:[ORION]MFPERS.AIJ;1
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-LOGRECOVR, 2 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 2 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence
number needed will be 1
%RMU-I-LOGOPNAIJ, opened journal file DUA12:[ORION]MFPERS1.AIJ;1
%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed
%RMU-I-LOGRECOVR, 2 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence
number needed will be 2
%RMU-I-LOGOPNAIJ, opened journal file DUA13:[ORION]MFPERS2.AIJ;1
%RMU-I-AIJONEDONE, AIJ file sequence 2 roll-forward operations completed
%RMU-I-LOGRECOVR, 2 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
```

(continued on next page)

Example 8–10 (Cont.) Restoring and Recovering the EMP_INFO Storage Area

```
%RMU-I-LOGSUMMARY, total 6 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
%RMU-I-LOGSUMMARY, total 2 transactions ignored
%RMU-I-AIJSUCCE, database recovery completed successfully
%RMU-I-AIJGOODAREA, storage area DUAL:[ORION]EMP_INFO.RDA:1 is now consistent
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence
number needed will be 2
$
$ ! Verify your database to be sure it is intact, including
$ ! the restored and recovered SALARY_HISTORY storage area.
$
$ RMU/VERIFY DB_DISK:[MFPERS]MF_PERSONNEL
```

For more examples and discussion of automatic and manual recovery, see Section 9.9 and Section 9.12.

A request to restore a storage area not included in the backup file results in an error, as shown in Example 8–11. The resulting database, typically the affected storage area, is unusable until the affected storage area is properly restored.

Example 8–11 Error If You Try to Restore a Storage Area That Is Not Included in the Backup File

```
$ RMU/BACKUP /EXCLUDE=DEPARTMENTS MF_PERSONNEL PERS_BACKUP.RBF
$ RMU/RESTORE /NEW_VERSION /AREA PERS_BACKUP.RBF DEPARTMENTS
%RMU-W-AREAEXCL, The backup does not contain the storage area - DEPARTMENTS
```

When this restore operation is attempted on a usable database, it does complete, but the DEPARTMENTS storage area may be inconsistent if the Norecovery qualifier is specified or the backup operation was so old that automatic recovery is not performed. In Example 8–12, an RMU Verify command shows the error that results when you try to verify the database.

Example 8–12 Error If You Try to Verify the Database

```
$ RMU/VERIFY MF_PERSONNEL
%RMU-W-BADPROID, STORAGE AREA file DEPARTMENTS.RDA;5 a bad identifier
Expected "RDMSDATA", found
%RMU-W-BADDBPRO, This STORAGE AREA file does not belong to
MF_PERSONNEL;4 database
      found references to database.
%RMU-W-INVALFILE, inconsistent database file DEPARTMENTS.RDA;5
%RMU-F-INVBSFIL, inconsistent storage area file DEPARTMENTS.RDA;5
```

To correct this situation, find the backup file containing the last full backup file of the DEPARTMENTS storage area, and restore the DEPARTMENTS storage area as shown in Example 8–13.

Example 8–13 Restoring an Inconsistent Storage Area from a Backup File Containing That Storage Area

```
$ RMU/RESTORE /NEW_VERSION /AREA /ONLINE PERS_BACKUPALL.RBF DEPARTMENTS
```

Now verify that the DEPARTMENTS storage area is consistent and check the version number of the DEPARTMENTS storage area file, noting that it is incremented by 1 as shown in Example 8–14.

Example 8–14 Verifying the Database and Checking the Version Number of the Storage Area File

```
$ RMU/VERIFY MF_PERSONNEL
$
$ !No error message means the database has been successfully verified.
$
$ DIR DEPARTMENTS.RDA
Directory DUAL:[ORION]
DEPARTMENTS.RDA;6  DEPARTMENTS.RDA;5  DEPARTMENTS.RDA;4
Total of 3 files.
```

Because a backup file may be a partial one of the database, without all storage areas present, you must ensure that all storage areas of a database are restored and recovered when using the RMU Backup and RMU Restore commands to duplicate a database. Oracle Rdb recommends that you make full database backup files containing all storage areas on a regular basis and use this backup file for duplicating the database.

In addition, if your database is very large, you can devise a partial database backup strategy for different storage areas. In this case, you can restore a database by specific storage areas based on your partial backup strategy. Always use your last full backup file containing all storage areas for duplicating the database. If you use your partial backup strategy to duplicate the database, you must realize that all storage areas must be restored properly for the database to be usable again. Using a complete backup file containing all storage areas to duplicate your database is easier and makes good sense. Therefore, use the RMU Restore command with the Area and Online qualifiers only when you want to restore one or a few storage areas.

8.6.1 Performing an Online By-Area Restore Operation on One Storage Area

Assume that the read/write SALARY_HISTORY storage area in the mf_personnel database was last backed up on a Friday. To restore on line only this storage area, use the RMU Restore command shown in Example 8-15, using the last full and complete backup file (pers_full_rw.rbf) that contains all the mf_personnel database read/write storage areas. Users can access other database storage areas while this operation is being performed, but they cannot access the storage area being restored during this restore operation.

Example 8-15 Online By-Area Restore Operation of a Read/Write Storage Area

```
$ RMU/RESTORE/NEW_VERSION /AREA /ONLINE -  
_$_$ DBS_BACKUPS:[MFPERS]PERS_FULL_RW.RBF SALARY_HISTORY
```

8.7 Performing By-Page Restore Operations

The RMU Restore command permits you to restore one or more database pages on line without restoring the entire database. Users are locked from using those pages that are being restored and recovered. If after-image journaling is enabled, restored pages must be recovered to make the pages consistent. The following scenario shows how to restore and recover a corrupt page and display the corrupt page table (CPT) to monitor the progress.

Suppose you perform a full verify operation and receive the following error message:

```
$ RMU/VERIFY/ALL MF_PERSONNEL  
%RMU-W-PAGCKSBAD, area DEPARTMENTS, page 4  
contains an invalid checksum  
expected: 7949BB61, found: 7949AE61  
%RMU-E-CORRUPTPG, Page 4 in area DEPARTMENTS is marked as corrupt.  
$
```


This error indicates that page 4 in the DEPARTMENTS table is corrupt. This information is automatically logged in the CPT. Display the CPT to note the entry, as follows:

```
$ RMU/SHOW CORRUPT_PAGES MF_PERSONNEL
.
.
.
Entries for storage area DEPARTMENTS
-----

Page 4
- AIJ recovery sequence number is -1
- Area ID number is 5
- Consistency transaction sequence number is 0
- State of page is: corrupt
.
.
.
```

To make the page accessible to users again, restore page 4 from a full and complete backup operation of the database by using the Just_Corrupt qualifier. If after-image journaling is enabled, make sure all journals are available and on disk; if journals are backed up, you will have to recover them manually; otherwise, journals are applied automatically as part of the restore operation.

```
$ RMU/RESTORE/LOG/AREA MFPEERS_FULL.RBF DEPARTMENTS/JUST_CORRUPT
%RMU-I-RESTXT_04, Thread 1 uses devices DISK3$:
%RMU-I-LOGRESSST, restored storage area DISK3$:[MFPEERS]DEPARTMENTS.RDA;1
%RMU-I-LOGRESSST, restored storage area DISK3$:[MFPEERS]DEPARTMENTS.RDA;1
%RMU-I-RESTXT_05, rebuilt 0 space management pages
%RMU-I-RESTXT_06, restored 0 inventory pages
%RMU-I-RESTXT_07, rebuilt 0 logical area bitmap pages
%RMU-I-RESTXT_08, restored 1 data page
%RMU-I-AIJWASON, AIJ journaling was active when the database was backed up
%RMU-I-AIJREFUL, Recovery of the entire database starts with AIJ file
sequence 0
%RMU-I-AIJBADPAGE, inconsistent page 4 from storage area
DISK3$:[MFPEERS]DEPARTMENTS.RDA;1 needs AIJ sequence number 0
%RMU-I-LOGRECD, recovering database file DB_DISK:MF_PERSONNEL.RDB;1
%RMU-I-AIJAUTOREC, starting automatic after-image journal recovery
%RMU-I-LOGOPNAIJ, opened journal file DISK11$:[JOURNAL]JOURN_1.AIJ;1
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-LOGRECOVR, 2 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 1 transaction ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCE, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence
number needed will be 1
%RMU-I-LOGOPNAIJ, opened journal file DISK12$:[JOURNAL]JOURN_2.AIJ;1
```

```

%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed
%RMU-I-LOGRECOVR, 2 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCESS, database recovery completed successfully
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-LOGSUMMARY, total 4 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
%RMU-I-LOGSUMMARY, total 1 transaction ignored
%RMU-I-AIJSUCCESS, database recovery completed successfully
%RMU-I-AIJGOODPAGE, page 4 from storage area DISK3$:[MFPERS]DEPARTMENTS.RDA;1
is now consistent
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence
number needed will be 3
%RMU-I-AIJNOENABLED, after-image journaling has not yet been enabled

```

Because the journals are on disk and available, the restore operation also proceeded to automatically recover the database page to make it consistent and update the CPT to indicate the CPT is now empty and all areas are consistent.

However, suppose for some reason the journals were not available when you restored the database (journals were backed up) or you specified the Norecovery qualifier in the restore operation. Under these circumstances you must recover the database manually. In this situation, you should follow the restore and recovery progress by displaying the status of the page by inspecting the CPT.

```

$ RMU/RESTORE/LOG/AREA/NORECOVERY MFPERS_FULL DEPARTMENTS/JUST_CORRUPT
%RMU-I-RESTXT_04, Thread 1 uses devices DISK3$:
%RMU-I-LOGRESSST, restored storage area DISK3$:[MFPERS]DEPARTMENTS.RDA;1
%RMU-I-LOGRESSST, restored storage area DISK3$:[MFPERS]DEPARTMENTS.RDA;1
%RMU-I-RESTXT_05, rebuilt 0 space management pages
%RMU-I-RESTXT_06, restored 0 inventory pages
%RMU-I-RESTXT_07, rebuilt 0 logical area bitmap pages
%RMU-I-RESTXT_08, restored 1 data page
%RMU-I-AIJWASON, AIJ journaling was active when the database was backed up
%RMU-I-AIJRECFUL, Recovery of the entire database starts with AIJ file
sequence 0

```

Recovery starts with .aij file sequence number 0. Display the CPT to determine the current status of page 4 as follows:

```

$ RMU/SHOW CORRUPT_PAGES MF_PERSONNEL
.
.
.
Entries for storage area DEPARTMENTS
-----

```

Page 4

- AIJ recovery sequence number is 0
- Area ID number is 5
- Consistency transaction sequence number is 88
- State of page is: inconsistent

.
.
.

The CPT indicates that the status of page 4 is changed from corrupt to inconsistent and that AIJ recovery should begin with AIJ sequence number 0 and the consistency transaction sequence number (TSN) is 88. Manually recover the journal files. If the remaining journal files are available and on disk, recovery proceeds automatically after the first journal file is applied.

```
$ RMU/RECOVER/JUST_CORRUPT/ONLINE/LOG JOURN_1
%RMU-I-AIJBADPAGE, inconsistent page 4 from storage area
DISK3$:[MFPERS]DEPARTMENTS.RDA;1 needs AIJ sequence number 0
%RMU-I-LOGRECDB, recovering database file DB_DISK:MF_PERSONNEL.RDB;1
%RMU-I-LOGOPNAIJ, opened journal file DISK11$:[JOURNAL]JOURN_1.AIJ;1
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-LOGRECOVR, 2 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 1 transaction ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCESS, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence
number needed will be 1
%RMU-I-AIJAUTOREC, starting automatic after-image journal recovery
%RMU-I-LOGOPNAIJ, opened journal file DISK12$:[JOURNAL]JOURN_2.AIJ;1
%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed
%RMU-I-LOGRECOVR, 2 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCESS, database recovery completed successfully
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-LOGSUMMARY, total 4 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
%RMU-I-LOGSUMMARY, total 1 transaction ignored
%RMU-I-AIJSUCCESS, database recovery completed successfully
%RMU-I-AIJGOODPAGE, page 4 from storage area
DISK3$:[MFPERS]DEPARTMENTS.RDA;1 is now consistent
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence
number needed will be 3
%RMU-I-AIJNOENABLED, after-image journaling has not yet been enabled
```

A display of the CPT indicates that all storage areas are now consistent, as follows:

```
$ RMU/SHOW CORRUPT_PAGES MF_PERSONNEL
.
.
.
Corrupt page table is empty.
.
.
.
All storage areas are consistent.
.
.
.
```

If you must manually apply each backed up journal file, both the informational messages from each recovery operation and the CPT indicate which AIJ sequence number to apply next and the last committed TSN for each page. This information guides you toward making your database consistent and accessible again by directing you to apply journals in a specific order. See the *Oracle RMU Reference Manual* for additional examples.

When the restore and recovery operations are complete and if journaling is now disabled, you should enable journaling again and back up your database. Doing this will ensure that you can restore and fully recover your database should this become necessary.

8.8 Restoring Only the Root File from a Database Backup File

If the database root (.rdp) file becomes corrupt as indicated by a verify operation, or if the database root file is accidentally deleted or lost due to a disk failure, normally you must restore the entire database to use it again. This task takes time if you have a very large database. An alternative is to restore just the database root file from a recent backup file by using the RMU Restore Only_Root command.

The RMU Restore Only_Root command permits you to:

- Restore only the .rdp file and optionally specify a new file and directory specification for the .rdp file.
- Restore only the .rdp file and optionally update the .rdp file and change database-wide parameters (number of cluster nodes, number of users, global buffer parameters, local buffer parameters, and the mode for opening the database).

- Restore only the root file and optionally update the .rdb file with the exact file locations and versions for each specified storage area (.rda) and snapshot (.snp) file, update each of four storage area parameters (page size, SPAM threshold values, setting the write-once and read/write attribute, and enabling or disabling space area management (SPAM) pages), and update one .snp file parameter (file allocation size).
- Restore only the .rdb file and optionally set the next TSN and commit sequence number (CSN) values to values equal to or greater than the values recorded in the backup file.
- Restore only the .rdb file and optionally initialize all TSN values in the database to the value 0.
- Restore only the .rdb file and optionally enable or disable after-image journaling, or continue after-image journaling but start a new .aij file.
- Restore only the .rdb file from tape, and optionally specify the set of tape labels on which the backup file resides and, if the initial tape volume is already mounted, whether it should be rewound before being read.

Caution

If you plan to use the RMU Restore Only_Root command for restoring an .rdb file, plan your use very carefully. Misuse of this command, such as specifying incorrect .rda and .snp file information during a restore only-root operation, can result in a corrupt database if the .rdb file is incorrectly updated. For example, if the .rdb file either no longer points to the set of correct database files including correct version numbers of these files, the database is unusable. In addition, if you reset the database TSN and CSN values and have after-image journaling enabled, you must create a new .aij file and immediately back up your database.

An .aij file containing discontinuous TSN and CSN numbers (that is, a gap between the last and next TSN and CSN values) or a gap between TSN and CSN numbers between two successive .aij files cannot be rolled forward beyond the point of discontinuity, making your database inconsistent. Refer to the *Oracle RMU Reference Manual* for further considerations and restrictions when using this command. Be prepared to restore your entire database should your database become corrupt or inconsistent from misuse of this command.

The following considerations and restrictions apply when using the RMU Restore Only_Root command:

- Be sure to correctly restore the .rdb file so that all .rdb file pointers are consistent with the actual set of existing database files.
- Be sure the database backup file is recent.

There are either no known differences between the database backup (.rbf) file and the missing root file or the differences are relatively few and are well known. Both situations pose the least risk in making the database usable again and both are relatively easy situations from which to recover. All differences must be resolved. You must determine what these differences are and determine how the .rdb file will be updated to resolve each difference. Some differences can be resolved by restoring the .rdb file and by specifying specific qualifiers and parameter values to update the .rdb file. Other differences may require that the .rdb file be updated manually by using the SQL interface after the .rdb file is restored. Finally, some differences may offer no alternative but to restore the entire database to make it usable again.

Note

If after-image journaling is enabled, you cannot roll forward the .aij file to update the .rdb file with the most current database-wide parameter and storage area parameter changes that were logged in the .aij file since the last backup operation. Rolling forward the .aij file makes your .rdb file inconsistent with the area files, thereby making your database inconsistent. The only alternative is to perform the restore-only root operation and manually update the .rdb file by using the SQL ALTER DATABASE statement.

This procedure makes the required changes to any of the database-wide parameters and storage or snapshot area parameters that were made or that occurred since the last backup operation and that could not be changed by using the restore only-root operation. Under some circumstances (for example, extending an area file) you cannot update the .rdb file manually to indicate that an area file extension has occurred. This is an instance in which affected areas must be restored and recovered to make the database usable again.

-
- The number of storage areas has not changed since the last backup operation.

If a storage area has been deleted then you must restore the entire database to make it usable again.

- You must perform a complete and full backup operation of your database immediately following a successful restore-only root operation.

If after-image journaling is enabled from the restore-only root operation, a new .aij file is created, or after-image journaling is being continued, a new .aij file must be created. The combination of the new backup file and the new .aij file provides a consistent point from which to roll forward should subsequent recovery be necessary.

- You must correctly reset the TSN and CSN values in your restored .rdb file if they are different from the values stored in the backup file.

If the TSN and CSN values are set to a value lower than the value stored in the original .rdb file, the database becomes corrupt and may return incorrect data or cause application failures. Also, if the TSN and CSN values are not set to the exact next TSN and CSN values that were stored in the original .rdb file, there will be a discontinuity in the journaling if after-image journaling is enabled. If you permit these values to increment by the default of one million (when you specify the Set_Tsn qualifier without specifying the TSN and CSN values), you must perform a full and complete backup operation for the reasons mentioned previously.

From this list of restrictions and considerations for using the RMU Restore Only_Root command, you should realize that successful use of this command requires some prior planning. To ensure success in using the RMU Restore Only_Root command, do the following tasks:

- Obtain a recent backup file of your database and regularly back up your database after modifying database-wide, storage and snapshot area parameters.
- Obtain a very recent output file by using the Options=Debug qualifier that contains the contents of the .rdb file header before the .rdb file was lost, so that you can compare its contents with the contents of your most recent backup file.
- Enable after-image journaling so you can check the .aij file for the highest TSN and CSN numbers of the last and most recently committed transaction from which you can determine what the next TSN and CSN values should be.

Making the database usable again with the restore-only root operation depends on making the restored .rdb file current with the contents of the .rdb file that was lost and consistent with the storage and snapshot area information. The difficulty depends on how much the lost .rdb file's contents actually differ from that of the most recent .rbf file. Differences can be estimated and gauged by comparing the contents of the most recent output file of the .rdb file header with the contents of most recent .rbf file. The following list describes various scenarios, from best case to worst case, that are possible when comparing a copy of the lost .rdb file's contents with that of the most recent .rbf file:

- There are no differences.

The best case is that there are no differences between the contents of the header portions of these two files. That is, no transactions have occurred to increment the TSN and CSN numbers in the .rdb file since the database was last backed up. Therefore, you need to restore only the .rdb file to make the database usable again and at the same time reset the TSN and CSN values to the values indicated in the .rbf file. If after-image journaling is enabled, display the contents of the .aij file and search for the TSN and CSN values that are higher than the next TSN and CSN values in the .rbf file to confirm this.

- Only the TSN and CSN values are different.

The next best case is that there was some transaction activity and only the TSN and CSN values are different. Now you must either estimate how much transaction activity occurred so you can estimate what the next TSN and CSN values should be and set these values either to exactly what they should be, to a little higher than they were, or to a much higher value if you cannot make a reasonable estimate. You can check the contents of your .aij file for the CSN and TSN value for the last committed transaction. This assumes that there are no outstanding recovery-unit journal (.ruj) files to be recovered by one or more database recovery (DBR) processes if the database failed due to a system problem. Again, you need to restore only the .rdb file to make the database usable again, reset the TSN and CSN values to new higher values, and if after-image journaling is enabled, create a new .aij file. A backup operation of the database is required immediately after restoring the .rdb file to ensure that the database can be rolled forward from a point that is consistent with this most recent .rbf file, should the database need to be restored again in the near future.

- The TSN and CSN values are different, plus some database-wide, or storage area, or snapshot parameters are different and can be reset by using the restore-only root command qualifiers.

Next in the order of best case scenarios is that you must update some additional parameter settings for the database, storage, and snapshot areas, and there are qualifiers for doing this operation. You might have gathered this information from keeping an up-to-date activity log of changes you have made to database, storage, or snapshot areas along with a copy of the most recent debug output display of the .rdb file header that shows the current settings before the .rdb file was lost, along with the next TSN and CSN values. You need to reset only these parameters, using the available qualifiers for the restore-only root operation.

- The TSN and CSN values are different, plus many database-wide, storage area, or snapshot parameters are different and can be reset only by using the SQL interface.

Next in increasing difficulty is the scenario in which you must reset some database, storage, or snapshot area parameters that can only be reset by using the SQL ALTER DATABASE statement.

- The TSN and CSN values are different, plus some database-wide, storage area, or snapshot parameters are different, and none of these parameters can be reset manually.

An even more difficult case is that there is some parameter or group of parameters that have changed but cannot be reset by any available interface or set of tools. For example, you might discover that a file extended and that you cannot manually update this kind of information in the .rdb file. Because this represents a potential inconsistency between your restored .rdb file and the area files that cannot be corrected, your only alternative is to restore and recover the area to make the area usable again.

- The disk containing the .rdb file becomes inoperable while there are active updates in the database.

Finally, an even worse case is if the .rdb file is completely inaccessible, .ruj file recovery cannot be performed, and any or all of the storage areas are inconsistent. Restoring the .rdb file and resolving all the .rdb file differences still does not permit .ruj file recovery because all the information needed for controlling the recovery is in the lost .rdb file. The only way to recover from this situation is to restore and recover the entire database.

In an extremely heavy transaction environment over a considerable period of time, your database TSN and CSN numbers may approach the maximum allowable value of 4,294,967,295. Before this maximum allowable value is reached, you should initialize all TSN and CSN values to zero, create a new .aij file if after-image journaling is enabled, and immediately back up your

database. No warning message displays if you approach or reach this value, and the values for this parameter do not automatically reset themselves to zero but cause database activity to cease until these values are reset to zero. Once you reset these values to zero, you are immediately required to back up your database, and start a new .aij file if after-image journaling is enabled, to provide a consistent point to roll forward from should subsequent recovery be necessary.

Example 8–16 shows how to check the current value for the TSN and CSN, to restore only the .rdb file and to initialize the TSN and CSN values to zero, to create a new .aij file if after-image journaling is enabled, to check that the values are reset, and then, immediately following the restore-only root operation, to back up the database.

Example 8–16 Performing a Restore-Only Root Operation and Initializing the TSN and CSN Values to Zero

```

$ ! Show the current next CSN and TSN values and note that the maximum
$ ! allowable value of 4,294,967,295 is very close.
$
$ RMU/DUMP/HEADER /OPTION=DEBUG DB_DISK:[MFPERS]MF_PERSONNEL
.
.
.
RWROOT @7FDDDF77C          NEXT_CSN = 4294967280.          NEXT_TSN = 4294967290.
.
.
.
$
$ ! Restore only the .rdb file and initialize the TSN and CSN values
$ ! to zero and start a new .aij file.
$
$ RMU/RESTORE/ONLY_ROOT /INITIALIZE_TSNS -
_$ DBS_BACKUPS:[MFPERS]MFPERS_FULLBKUP.RBF
_$ /AFTER_JOURNAL=MFPERS.AIJ
$
$ ! Check that the current next TSN and CSN values are reset to zero.
$
$ RMU/DUMP/HEADER /OPTION=DEBUG DB_DISK:[MFPERS]MF_PERSONNEL
.
.
.
RWROOT @7FDDDF77C          NEXT_CSN = 0.          NEXT_TSN = 0.
.
.
.

```

(continued on next page)

Example 8–16 (Cont.) Performing a Restore-Only Root Operation and Initializing the TSN and CSN Values to Zero

```
$  
$ ! Immediately back up the database.  
$  
$ RMU/BACKUP DB_DISK:[MFPERS]MF_PERSONNEL -  
_ $ DBS_BACKUPS:[MFPERS]FULL_BACKUP.RBF
```

Assume that your .rbf file is not too recent and that since you last backed up your database, you made the RESUME_LISTS storage area containing list data a write-once storage area and then moved it to a write-once, read-many (WORM) optical device. After completing this operation, you lost your .rdb file. Your .rbf file knows nothing of these changes. For this reason, you want to restore only the .rdb file and update the .rdb file with the current location of the RESUME_LISTS storage area, the correct version of the resume_lists.snp file, and reset the WORM attribute for the RESUME_LISTS storage area in the .rdb file. Example 8–17 shows how to check the file locations and version numbers for the RESUME_LISTS storage area and .snp file, restore only the .rdb file while specifying the new location for the RESUME_LISTS storage area and the correct version for the resume_lists.snp file, and reset the WORM attribute for the RESUME_LISTS storage area.

Example 8–17 Performing a Restore-Only Root Operation and Specifying .rda File, WORM, and .snp File Parameters

```
$ ! Check the location and version number of the RESUME_LISTS storage area  
$ ! on the WORM optical disk device ODA0 and the resume_lists.snp file on  
$ ! disk device DISK1.  
$  
$ DIR WORM1:[MFPERS]RESUME_LISTS.RDA;1  
Directory WORM1:[MFPERS]  
  
RESUME_LISTS.RDA;1  
  
Total of 1 file.  
$  
$ DIR DISK1:[MFPERS]RESUME_LISTS.SNP  
Directory DISK1:[MFPERS]  
  
RESUME_LISTS.SNP;2  
  
Total of 1 file.
```

(continued on next page)

Example 8-17 (Cont.) Performing a Restore-Only Root Operation and Specifying .rda File, WORM, and .snp File Parameters

```
$
$ ! Restore only the .rdb file and specify the actual location of the
$ ! resume_lists.rda file, the correct version of the resume_lists.snp
$ ! file, reset the WORM attribute for the RESUME_LISTS storage area
$ ! in the .rdb file, and create a new .aij file.
$
$ RMU/RESTORE/ONLY_ROOT DBS_BACKUPS:[MFPERS]MFPERS_FULLBKUP.RBF -
_$ RESUME_LISTS /FILE=WORM1:[MFPERS]RESUME_LISTS.RDA /WORM
_$ /SNAPSHOT=(FILE=DISK1:[MFPERS]RESUME_LISTS.SNP;2)
_$ /AFTER_JOURNAL=MFPERS.AIJ
$
$ ! Display the updated .rdb file to show the changes made.
$
.
.
.
Storage area RESUME_LISTS
  Area ID number is 9
  Filename is "WORM1:[MFPERS]RESUME_LISTS.RDA;1"
.
.
.
  Status...
  Area has the WORM attribute
.
.
.
Snapshot area for storage area RESUME_LISTS
  Area ID number is 19
  Filename is "DISK1:[MFPERS]RESUME_LISTS.SNP;2"
.
.
.
$
$ ! Verify the .rdb file; no errors returned indicates a sound .rdb file.
$
$ RMU/VERIFY/ROOT DB_DISK:[MFPERS]MF_PERSONNEL
$
```

(continued on next page)

Example 8–17 (Cont.) Performing a Restore-Only Root Operation and Specifying .rda File, WORM, and .snp File Parameters

```
$
$ ! Check that the next TSN and CSN values are incremented by
$ ! default to 1 million plus the value in the backup file.
$
$ RMU/DUMP/HEADER /OPTION=DEBUG DB_DISK:[MFPERS]MF_PERSONNEL
.
.
.
RWROOT @7FDDDF77C          NEXT_CSN = 1000072.          NEXT_TSN = 1000088.
.
.
.
$
$ ! Immediately back up the database.
$
$ RMU/BACKUP DB_DISK:[MFPERS]MF_PERSONNEL -
_ $ DBS_BACKUPS:[MFPERS]FULL_BACKUP.RBF
```

If you plan to use the RMU Restore Only_Root command to restore the .rdb file, be certain that your database backup (.rbf) file is recent or at the very least that you back up your database following changes to any database-wide parameters, storage area parameters, and .snp file parameters. You should also set up a regular procedure to save the contents of the debug form of the .rdb file header into an output file. You do this so you can obtain the most current copy of the database parameter settings in case you lose the .rdb file. These parameter settings can be compared with the contents of the .rbf file to find any differences, such as the next TSN and CSN values before the .rdb file was lost.

Using this strategy to find differences between the .rbf file and the most recent copy of the .rdb header allows you to track minor differences, such as different next TSN and CSN values. Finding these small differences permits you to restore the .rdb file fairly quickly and successfully. As mentioned previously, the more differences there are between the last .rbf file and the last display of the .rdb header (before you lost the .rdb file), the more difficult it is to make the restored .rdb file consistent with the .rda and .snp files, if a restore-only root operation is even possible.

8.9 Restoring a Database Directly from Tape

There are two ways of restoring .rbf files directly from tape. Each complements the backup methods described in Section 7.13.

- Restore the database from one tape device, from one or more tape volumes, loaded sequentially.
- Restore the database from multiple tape devices, from multiple tape volumes.

Use a single tape drive for restoring small databases that all fit on one backup tape or when there is only one tape device on the system. Use multiple tape drives for restoring moderate to very large backed up databases and when two or more tape devices are part of the system. The restore procedure for both is generally the same; the differences are in how many devices are requested to be allocated, mounted, dismounted, and deallocated in each operating system command, and how many devices are specified in the RMU command.

8.9.1 Using a Single Tape Drive

When you are using only one tape, you can restore a database directly from tape by following this procedure. When using two or more tapes on one tape drive, you must dismount/mount the tapes to load the first tape for the optional step of verifying the backup file before restoring the database. The procedure steps follow:

1. Allocate the tape drive.
2. Mount the tape.
3. Perform the restore operation.
4. Dismount the tape.
5. Deallocate the tape drive.
6. Verify the database.

This procedure (shown in Example 8–18) assumes that you have deleted any previous version of the database from your system.

Example 8–18 Using a Single Tape Drive for a Full Restore Operation

```
$ ALLOCATE $111$MUA0:
$
$ MOUNT/FOREIGN $111$MUA0:
$-
$ RMU/RESTORE /LOG /REWIND /LABEL=BACK01 $111$MUA0:PERS_FULL_MAR30.RBF
$
$ ! The Rewind qualifier rewinds the tape to the beginning.
$ ! The volume label is that label name specified in the
$ ! Label qualifier.
$
$ DISMOUNT $111$MUA0:
$
$ DEALLOCATE $111$MUA0:
$
$ RMU/VERIFY DB_DISK:[MFPERS]MF_PERSONNEL
$
$ ! Verify your restored database to be sure it is intact.
```

If your database is large, you may need to use more than one tape on your single tape device. In this case, you are prompted to mount additional tapes as each is read and restored up until the restore operation is complete.

8.9.2 Using Multiple Tape Drives

You can restore a database directly from two or more tapes on two or more tape devices with the following procedure:

1. Allocate each tape drive.
2. Mount the first tape.
3. Perform the restore operation.
4. Dismount the last tape.
5. Deallocate each tape drive.
6. Verify the database (this is optional).

This procedure (shown in Example 8–19) assumes that you have deleted any previous version of the database from your system.

Example 8–19 Using Multiple Tape Drives for a Full Restore Operation

```
$ ALLOCATE $111$MUA0:
$ ALLOCATE $222$MUA1:
$
$ MOUNT/FOREIGN $111$MUA0:
$
$ RMU/RESTORE /LOG /REWIND /LABEL=(BACK01,BACK02) -
_$ $111$MUA0:PERS_FULL_MAR30.RBF, $222$MUA1:
$
$ DISMOUNT $111$MUA0:,$222$MUA1:
$
$ DEALLOCATE $111$MUA0:
$ DEALLOCATE $222$MUA1:
$
$ RMU/VERIFY DB_DISK:[MFPERS]MF_PERSONNEL
```

In Example 8–19, the restore operation could use two different HSC devices in a multithreaded operation and restore the database to a similar configuration from which it was backed up. This is accomplished by simultaneously reading input through two different HSC devices and concurrently writing output to disks until the restore operation is complete. The information that follows explains how this is done.

When you have multiple tapes to restore, you should be sure to use the Journal qualifier in the backup operation to improve tape performance in the restore operation. The contents of the .jnl file are used in the restore operation. It contains a description of the backup command, the backup operation, the backup file name, the root file specification, the drive specification; the journals and areas backed up, their size, the high TSN and CSN for each journal, the high TSN for each area, the area ID and page size, the AIJ sequence number to use to recover the area; the snapshot area ID, its file specification and page size; and the relative volume used to back up each area and its starting location. To display the contents of a .jnl file, use the following command:

```
$ RMU/DUMP/BACKUP_FILE MFPERS1.RBF/JOURNAL=BKUP_JOURNAL.JNL/OPTION=JOURNAL
```

See the *Oracle RCU Reference Manual* for an example of the contents of a .jnl file.

For large restore operations, if you backed up your database by using the Loader_Synchronization and Journal qualifiers, you must preload your tape volumes into the loaders or stackers in the same order in which they were backed up to achieve an unattended, concurrent restore operation. Oracle RCU uses the Journal qualifier to read the contents of the .jnl file to determine what the tape labels are and the order in which to expect to read the tapes.

When you use the Journal qualifier you do not need to specify the tape labels with the Label qualifier.

When you specify the Loader_Synchronization qualifier with the RMU Restore command, all concurrent tape operations must conclude before reading the next set of tape volumes. This ensures that tapes can be loaded and read in the loaders or stackers in the order required by the restore operation as specified by the Label qualifier or the .jnl file. You must also specify the Volumes qualifier and the total number of volumes that must be restored to make this a concurrent restore operation. If a mistake is made while loading the tape volume in the loader or stacker, Oracle RMU prompts the operator for tape disposition as described in Section 7.13.4.

For very large backup and restore operations requiring many tape volumes, you should consider using a third-party storage library system to manage the tape labeling, preloading, and restoring of your database.

To control the level of concurrency used by the restore operation, you can use the Master qualifier to specify which drives are masters and which are slaves. The control is particularly useful when you also use the Loader_Synchronization qualifier. If you specify all drives as master drives, the RMU restore operation reads from all drives concurrently. The drives not designated as masters automatically become slave drives.

All master drives are read from concurrently and when these tape volumes are all read, then the tape volumes in the first set of slave drives are read; when the first set of slave drive tape volumes are all read, then the tape volumes in the second set of slave drives are read, and so forth. When the tape volumes in the last slave drive set are all read, then tape volumes in the master drive set are read and so forth until the database is completely restored. See the *Oracle RMU Reference Manual* for an example of this operation.

If after-image journaling was enabled when you backed up the database and journal files are available and on disk, recovery automatically follows the restore operation; if journal files are backed up or you want to restore incrementally, you should specify the Norecovery qualifier in the restore operation, then manually apply all backed up journal files. If there are any on-disk journal files, these will automatically be recovered following the rolling forward of the last backed up journal file in the recovery sequence.

8.10 Exceeded Quotas During a Database Restore or Backup Operation

If you receive an exceeded quota error message during either a restore or backup operation as shown here for a restore operation, you need to check your DIOLM and ASTLM parameter values:

```
-SYSTEM-F-EXQUOTA, exceeded quota
%RMU-F-FATALERR, fatal error on RESTORE
```

The DIOLM and ASTLM parameter values may be too low for the particular database or the type of backup operation (multithreaded versus single-threaded), and you need to set these parameter values higher. As a general rule, use the following algorithm for estimating the DIOLM parameter value and add 12 to the DIOLM calculated estimate to determine the value for the ASTLM parameter.

$$DIOLM = (3 * \text{Number of storage areas} + 5 * \text{Number of output threads})$$

$$ASTLM = DIOLM + 12$$

The number of threads can be determined from inspecting the log of the failed restore operation. If you received the exceeded quota error message during a backup operation, inspect the backup log to determine how many output threads there were. Find the highest output thread value in the log, and use that value to estimate the DIOLM and ASTLM parameter values for both the backup and restore operations.

The estimated value for the DIOLM parameter is higher for the backup operation if you are using multithreaded backup operations to tape, so parameter estimates should be based on this kind of operation and configuration. Therefore, estimate the DIOLM parameter value for the database on your system with the highest number of storage areas, and for the multithreaded tape backup operation (if your system is so configured), to ensure that the DIOLM and ASTLM parameter values are properly set for backup and restore operations.

8.11 Modifying Database Characteristics During a Restore Operation

During a restore operation, you can modify several database characteristics and storage area characteristics.

You can modify the following database characteristics when restoring a database:

- Restore the .rdb file access control list (ACL) from the .rbf file by using the Acl qualifier or use the Noacl qualifier to prevent this.
- Enable or disable after-image journaling, or if already enabled, create a new .aij file by using the After_Journal or the Noafter_Journal qualifier.
- Specify a journal options file with the Aij_Options qualifier to do any of the following:
 - Enable or disable after-image journaling
 - Reserve more journal slots
 - Allocate journal space
 - Specify extent size
 - Specify manual or automatic AIJ backup operations
 - Enable or disable journal overwrite
 - Specify a shutdown timeout value
 - Enable or disable notification
 - Enable or disable a journal file cache
 - Add journals
 - Specify the file name, a backup file name, the allocation size, and extent size for each journal added

See the *Oracle RMU Reference Manual* for more information.

- Retain the original journal files if you do not specify an After_Journal, Noafter_Journal, Aij_Options, or Noaij_Options qualifier
- Set the maximum number of cluster nodes with the Nodes_Max qualifier.
- Set the maximum number of database users with the Users_Max qualifier.
- Change the default global buffer parameters with the Global_Buffers qualifier.

- Change the default local buffer parameters with the `Local_Buffers` qualifier.
- Change the mode for opening the database with the `Open_Mode` qualifier.

You can modify the following storage area characteristics when restoring a database:

- Set the space area management (SPAM) threshold values for storage areas with mixed page format with the `Thresholds` qualifier.
- Set the number of blocks per page with the `Blocks_Per_Page` qualifier.
- Enable or disable SPAM pages for a specified storage area with the `Spams` or `Nospams` qualifier.
- Enable or disable the WORM attribute for a specified storage area with the `Worm` or `Noworm` qualifier.
- Change a read/write or write-once area to a read-only area with the `Read_Only` qualifier or change a read-only or write-once area to a read/write area with the `Read_Write` qualifier.
- Change the automatic file extension attribute for a storage area with the `Extension` qualifier to disable it (`Extension=Disable`) or enable it (`Extension=Enable`).
- Change the page size for a storage area with the `Blocks_Per_Page` qualifier.
- Change the `.snp` file allocation with the `Snapshot=Allocation` qualifier and a new file specification with the `Snapshot=File` qualifier.
- Change the file specification for a storage area with the `file` qualifier.

Modifying these database characteristics during an RMU Restore operation, compared to using an SQL EXPORT and IMPORT operation, has one immediate benefit—the RMU Restore operation may be several times faster because it is not restructuring the database. If these are the only database characteristics you want to modify, this may be a preferred method, especially if you are also planning to restore your database. See the *Oracle Rdb7 Guide to Database Performance and Tuning* for a comparison of options and the different ways that these options can be carried out. (The SQL ALTER DATABASE statement is the preferred way to modify the after-image journal characteristics and enable its use.)

An even better method to modify database characteristics is to unload the database, recreate it specifying changed parameters, and load it again. Use the RMU Extract command to create the data definition language (DDL) procedure to recreate the database and unload and load procedures for each table. See

the *Oracle RMU Reference Manual* for more information on using the RMU Extract command.

Modifying these two storage area characteristics (SPAM threshold values for mixed page format storage areas and page size) with a restore operation is not as useful as an export and import operation or a rebuild (unload it, re-create the database, and reload it using RMU Extract procedures) of the database for the following reasons:

- Your storage area is not reorganized.
Old data remains where it was, based on the previous page size and SPAM thresholds. Rows or records, if fragmented, are still fragmented.
- Only new data is stored based on these new page sizes and SPAM threshold values.

Over time, as data is archived and replaced by new data, this operation has a gradual impact on reorganizing your database. However, the preferred reorganizing tool is the use of the SQL EXPORT and IMPORT statements for certain changes (buffer size) and the SQL ALTER DATABASE statement for other database changes. When this operation (SQL EXPORT or IMPORT or SQL ALTER DATABASE) is complete, the entire database, both old and new data, is reorganized based on the new specifications.

Caution

The RMU Restore command has parameter qualifiers with positional semantics. When positioned *before* the first parameter, the area parameter qualifiers, `Blocks_Per_Page` and `Thresholds`, are global. When used globally, the specified parameter qualifiers and values affect all storage areas upon command execution. When positioned *after* each instance of a File or Area qualifier, the area parameter qualifiers, `Blocks_Per_Page` and `Thresholds`, are local. When used locally, the specified parameter qualifiers and values affect only the specified storage area and file name. See the *Oracle RMU Reference Manual* for more information on the positional semantics of Oracle RMU command qualifier parameters.

Care should be exercised if you intend to use the RMU Restore command to specify different SPAM thresholds and page sizes for specific storage areas.

For reorganizing data in storage areas when export and import operations take too long to complete, use the RMU Unload and RMU Load commands, especially for large databases. You can also use the REORGANIZE clause in the SQL ALTER DATABASE statement to move rows among storage areas. For example, use this clause if you want to move rows across two or more storage areas if a table and index is partitioned, or move rows among pages in the same storage area to pages on or closer to the new target pages if there is space. However, for both of these operations, you cannot modify any of the database characteristics.

8.11.1 Modifying After-Image Journaling Characteristics

After-image journaling characteristics can be modified during a restore operation by using either the After_Journal or the Aij_Options qualifier. The After_Journal qualifier is maintained for compatibility with versions of Oracle Rdb prior to V6.0 and can only be used to create a single extensible journal file. The Aij_Options qualifier can be used to create both a single extensible journal file or multiple fixed-size journal files. The Aij_Options qualifier uses a journal options file in which you specify the journaling configuration you want to use for your database. The format for the Aij_Options options file is described later in this section.

By default, if you restore a database and do not specify an After_Journal, Noafter_Journal, Aij_Options, or Noaij_Options qualifier and automatic recovery can complete, the original journal state (enabled or disabled) is restored from the backup file and the restored database tries to reuse the .aij file or files.

If you restore a database and specify the After_Journal qualifier with no file name, a new version of the single extensible after-image journal (.aij) file with the same name as the journal that is active at the time of the database backup operation is created (as shown in Example 8-20), if after-image journaling was enabled when you backed up the database.

Example 8-20 Modifying After-Image Journaling Characteristics During a Restore Operation

```
$ RMU/RESTORE/NEW_VERSION/AFTER_JOURNAL/LOG DBS_BACKUPS:[MFPERS]PERS_FULL.RBF
.
.
.
%RMU-I-LOGFILACC, created after-image journal file AIJ_DISK:[MFPERS]MF_PERSONNEL.AIJ;12
```

In Example 8-20, the RMU Restore command used the default .aij file name in the backup file to create a new version of mf_personnel.aij during the restore operation.

In either of the previous restore operation examples, if for some reason automatic recovery cannot complete (for example, journals are backed up or unavailable, such as on a disk that is off line), journaling is disabled and you receive a warning message similar to the following:

```
%RMU-I-AIJWASON, AIJ journaling was active when the database was backed up
%RMU-I-AIJRECFUL, Recovery of the entire database starts with AIJ file
sequence 0
%RMU-I-AIJRECBEG, recovering after-image journal "state" information
%RMU-I-AIJRSTAVL, 0 after-image journals available for use
%RMU-I-AIJISOFF, after-image journaling has been disabled
%RMU-I-AIJRECEND, after-image journal "state" recovery complete
%RMU-W-USERECCOM, Use the RMU/RECOVER command. The journals are not available.
```

If you want to disable after-image journaling during a restore operation, specify either the `Noafter_Journal` qualifier or the `Aij_Options` qualifier with no journal file option name specified with the RMU Restore command, as shown in Example 8–21.

Example 8–21 Disabling After-Image Journaling During a Restore Operation

```
$ RMU/RESTORE/NEW_VERSION/NOAFTER_JOURNAL -
_ $ DBS_BACKUPS:[MFPERS]PERS_FULL.RBF

$ RMU/RESTORE/NEW_VERSION/AIJ_OPTIONS -
_ $ DBS_BACKUPS:[MFPERS]PERS_FULL.RBF
```

The `Noafter_Journal` qualifier or the `Aij_Options` qualifier with no file specified has no effect if after-image journaling was not enabled for the database when it was backed up.

Use either the `After_Journal` qualifier or use the `Aij_Options` qualifier and specify a journal options file that contains a new journal file specification and enables journaling if you want to change the file specification and enable the `.ajj` file during a restore operation, as shown in Example 8–22.

Example 8–22 Changing the After-Image Journal File Specification During a Restore Operation

```
$ DEFINE/SYSTEM NEW_DISK $2$DUA12:[PERS.AIJ.MFPERS]
$ RMU/RESTORE/NEW_VERSION -
_ $ /AFTER_JOURNAL=NEW_DISK:MF_PERSONNEL.AIJ -
   PERS_FULL.RBF
```

The commands shown in Example 8–22 create an after-image journal file called `mf_personnel.ajj` in the device and directory pointed to by the `NEW_DISK` logical name.

To create a journal options file, use the `RMU Show After_Journal` command and specify an output file to capture the current journal configuration. Edit the output file to create your options file. For example:

```
$ RMU/SHOW AFTER_JOURNAL MF_PERSONNEL /OUTPUT=AIJ_CONFIG.OPT
JOURNAL IS DISABLED -
  RESERVE 1 -
  ALLOCATION IS 512 -
  EXTENT IS 512 -
  OVERWRITE IS DISABLED -
  SHUTDOWN_TIMEOUT IS 60 -
  NOTIFY IS ENABLED - ! Operators: OPER1
  BACKUPS ARE MANUAL -
  CACHE IS DISABLED
$
```

For example, to create three fixed-size journals, you might create a journal options file that contains the following configuration:

```
JOURNAL IS ENABLED -
  RESERVE 4 -
  ALLOCATION IS 512 -
  EXTENT IS 512 -
  OVERWRITE IS DISABLED -
  SHUTDOWN_TIMEOUT IS 60 -
  NOTIFY IS ENABLED - ! Operators: CENTRAL, CLUSTER
  BACKUPS ARE MANUAL -
  CACHE IS DISABLED
ADD JOURNAL JOURN_1 -
  FILE DISK11$:[JOURNAL]JOURN_1 -
  ALLOCATION IS 512 -
  BACKUP DISK21$:[BKUP_JOURNAL]BKUP_JOURN_1.AIJ;
ADD JOURNAL JOURN_2 -
  FILE DISK12$:[JOURNAL]JOURN_2 -
  ALLOCATION IS 512 -
  BACKUP DISK22$:[BKUP_JOURNAL]BKUP_JOURN_2.AIJ;
ADD JOURNAL JOURN_3 -
  FILE DISK13$:[JOURNAL]JOURN_3 -
  ALLOCATION IS 512 -
  BACKUP DISK23$:[BKUP_JOURNAL]BKUP_JOURN_3.AIJ;
```

Then, restore your database using this journal options file configuration as follows:


```

$ RMU/RESTORE/NEW_VERSION/LOG -
_ $ /AIJ_OPTIONS=AIJ_CONFIG.OPT -
_ $ DBS_BACKUPS:[MFPERS]PERS_FULL.RBF
%RMU-I-REXTXT_04, Thread 1 uses devices DB_DISK:
%RMU-I-REXTXT_18, Processing options file AIJ_CONFIG.OPT
.
.
.
%RMU-I-AIJWASON, AIJ journaling was active when the database was backed up
%RMU-I-AIJRECFUL, Recovery of the entire database starts with AIJ file sequence 1
%RMU-I-LOGCREAIJ, created after-image journal file DISK11$:[JOURNAL]JOURN_1.AIJ;1
%RMU-I-LOGCREAIJ, created after-image journal file DISK12$:[JOURNAL]JOURN_2.AIJ;1
%RMU-I-LOGCREAIJ, created after-image journal file DISK13$:[JOURNAL]JOURN_3.AIJ;1

```

See the *Oracle RMU Reference Manual* for more information about modifying the journal configuration when you perform a database restore operation.

8.11.2 Modifying SPAM Thresholds Values

You can modify the threshold values on your space area management (SPAM) pages for storage areas with mixed page format only during a restore operation by using the Thresholds qualifier. The Thresholds qualifier is a positional storage area qualifier that changes the storage area fullness percentage thresholds for use on SPAM pages. If you do not specify the Thresholds qualifier in an RMU Restore command, the original threshold values (stored in the .rdb file) are used. If the storage area pages are uniform, the thresholds are fixed; they cannot be modified and the Thresholds qualifier is in error. If you specify fewer than three values, the remaining values will be set at 100% for those values that were omitted. See *Oracle RMU Reference Manual* for examples of modifying the SPAM thresholds for storage areas with mixed page format.

8.11.3 Modifying Blocks per Page

You can use the Blocks_Per_Page qualifier to increase the size of storage area pages during a restore operation. By increasing the number of blocks per page in a particular area, you increase the amount of available space per page in that area. If your database is getting full, you can use the RMU Restore . . . /Blocks_Per_Page command to create new available space on each page in a storage area without interfering with row clustering. Existing rows are not touched in any way; if some are fragmented, they remain fragmented. But, if you subsequently modify a fragmented row in the newly restored database, it may be stored as an unfragmented row if there is now sufficient space for that row on the page.

If the page size of a storage area is already greater than or equal to the specified number of disk blocks, this qualifier is ignored.

The default, `Noblocks_Per_Page`, causes the original page sizes for a storage area to be retained from the backed up database.

8.11.4 Restoring List Storage Areas to WORM Optical or Read/Write Disk Devices

During a restore operation, you can restore read/write list storage areas to write-once, read-many (WORM) optical disk devices. When you do this, you must also set the Worm (write-once) attribute and disable the use of SPAM pages for the specified storage area and optionally set the `.snp` file allocation size to a smaller value if you want to conserve disk space. Use the `Worm`, `Nospams`, and `Snapshot=(Allocation=N)` qualifiers to set these parameters.

Only list storage areas with mixed page format can be restored to WORM optical disk devices because SPAM pages cannot be disabled for uniform page format storage areas. Disabling the use of SPAM pages is a requirement for restoring read/write storage areas to WORM optical disk devices.

You can also restore write-once list storage areas located on WORM optical disk devices to read/write disk devices. When you do this, you must also use the `Noworm` qualifier (read/write attribute), enable the use of SPAM pages for the specified storage area, and set the `.snp` file allocation size to a larger value if it was set previously to a small value to conserve disk space. Use the `Noworm`, `Nospams`, and `Snapshot=(Allocation=N)` qualifiers to set these parameters.

All three qualifiers, `Worm`, `Spams`, and `Snapshot=(Allocation=N)`, are positional storage area qualifiers that must be properly placed in the command line to guarantee the desired final result for the specified list storage area. See the *Oracle RCU Reference Manual* for more information on using positional qualifiers.

Example 8–23 shows how to restore a read/write list storage area that was originally on a read/write disk device to a WORM optical disk device, how to set the WORM attribute, how to disable the use of SPAM pages, and how to set the `.snp` file allocation to three pages for the associated storage area.

Example 8–23 Restoring a List Storage Area to a WORM Optical Disk Device from a Read/Write Disk Device

```
$ RMU/RESTORE/NEW_VERSION MFPERs_FULL.RBF -  
_$_ RESUME_LISTS/FILE=WORM1:[MFPERs]RESUME_LISTS.RDA /WORM /NOSPAMS  
_$_ /SNAPSHOT=(ALLOCATION=3)
```

Example 8–24 shows how to restore a write-once list storage area that was originally on a WORM optical disk device to a read/write disk device, how to set the NOWORM attribute, how to enable the use of SPAM pages, and how to set the .snp file allocation to 50 pages for the specified storage area.

Example 8–24 Restoring a List Storage Area to a Read/Write Disk Device from a WORM Optical Disk Device

```
$ RMU/RESTORE/NEW_VERSION MFPERS_FULL.RBF -  
_$_ RESUME_LISTS/FILE=DISK3:[MFPERS]RESUME_LISTS.RDA /NOWORM /SPAMS  
_$_ /SNAPSHOT=(ALLOCATION=50)
```

8.12 Performing Additional Tasks During a Restore Operation

During a restore operation, you can perform the following additional tasks:

- Use an options file.
- Create a duplicate database.
- Move database files to other devices.
- Move data dictionary information to other devices.
- Update the data dictionary.

8.12.1 Using an Options File to Restore a Database

When you must make many changes to your database during a restore operation and you want to record these changes and apply them to the restore operation, or if the length of your operating system Oracle RMU command line exceeds the maximum allowable number of characters, you can use an options file that contains all your specified changes. Then, reference the options file in the Oracle RMU command line by using the Options qualifier. The rules for using an options file follow:

- The options file contains storage area names, each of which is followed by the storage area qualifiers to be applied to that storage area. It has a default file type of .opt.
- Each storage area name is placed on its own line in the file. Do not separate the storage area names with a comma.
- You can use the storage area qualifiers `Blocks_Per_Page`, `File`, `Thresholds`, `Worm`, and `Spams` and the snapshot area qualifiers `Snapshot=(File=filename)` and `Snapshot=(Allocation=N)` in the options file.

- You can use the operating system line continuation character or the comment character in the options file.

Example 8–25 shows how to restore a database by using an options file. Assume that you have created SIGNATURES and PHOTOS storage areas containing the SIGNATURES and PHOTOS tables respectively, each of which contains list data (personnel signatures and ID photos) for the mf_personnel database. In this example, the SIGNATURES, and PHOTOS storage areas are restored to a WORM optical disk device. Each storage area is given the WORM attribute, each has the use of SPAM pages disabled, and each associated .snp file is allocated less space. In addition, the RESUME_LISTS storage area is restored to a read/write device, and because it has the read/write attribute the use of SPAM pages is enabled. The associated .snp file is allocated more space.

Example 8–25 Using an Options File to Restore a Database

```
$ TYPE WORM_CHANGES.OPT
SIGNATURES/FILE=WORM22:[MFPERS]SIGNATURES.RDA /WORM /NOSPAMS -
  /SNAPSHOT=(ALLOCATION=3)
PHOTOS/FILE=WORM22:[MFPERS]PHOTOS.RDA /WORM /NOSPAMS -
  /SNAPSHOT=(ALLOCATION=3)
RESUME_LISTS/FILE=DISK3:[MFPERS]RESUME_LISTS.RDA /NOWORM /SPAMS -
  /SNAPSHOT=(ALLOCATION=50)
$
$ RMU/RESTORE/NEW_VERSION /OPTIONS=WORM_CHANGES.OPT MFPERS_LISTS_FULL.RBF
```

All other database .rda and .snp files not specified in the .opt file are restored to the disk and directory specified in the .rbf file.

Example 8–26 shows how to use an .opt file to restore a database and how to change the .rda area and .snp file locations listed in the .rbf file. A new .ajj file is created as specified in the command line.

Example 8–26 Using an Options File to Relocate All Database Files to Restore a Database

```
$ TYPE AREA_CHANGES.OPT
RDB$SYSTEM      /FILE=DB_DISK12:[MFPERS]MF_PERS_DEFAULT -
                /SNAPSHOT=(FILE=DB_DISK12:[MFPERS]MF_PERS_DEFAULT)
MF_PERS_SEGSTR  /FILE=DISK13:[MFPERS]MF_PERS_SEGSTR -
                /SNAPSHOT=(FILE=DISK20:[MFPERS]MF_PERS_SEGSTR)
EMPIDS_LOW      /FILE=DISK14:[MFPERS]EMPIDS_LOW -
                /SNAPSHOT=(FILE=DISK19:[MFPERS]EMPIDS_LOW)
EMPIDS_MID      /FILE=DISK15:[MFPERS]EMPIDS_MID -
                /SNAPSHOT=(FILE=DISK 18:[MFPERS]EMPIDS_MID)
EMPIDS_OVER     /FILE=DISK16:[MFPERS]EMPIDS_OVER -
                /SNAPSHOT=(FILE=DISK17:[MFPERS]EMPIDS_OVER)
DEPARTMENTS    /FILE=DISK17:[MFPERS]DEPARTMENTS -
                /SNAPSHOT=(FILE=DISK16:[MFPERS]DEPARTMENTS)
SALARY_HISTORY  /FILE=DISK18:[MFPERS]SALARY_HISTORY -
                /SNAPSHOT=(FILE=DISK15:[MFPERS]SALARY_HISTORY)
JOBS           /FILE=DISK19:[MFPERS]JOBS -
                /SNAPSHOT=(FILE=DISK14:[MFPERS]JOBS)
EMP_INFO       /FILE=DISK20:[MFPERS]EMP_INFO -
                /SNAPSHOT=(FILE=DISK13:[MFPERS]EMP_INFO)

$ RMU/RESTORE /OPTIONS=AREA_CHANGES.OPT MFPERS_FULL.RBF -
_.$ /AFTER_JOURNAL=AIJ_DISK21:[MFPERS]MF_PERSONNEL.AIJ
```

See Section 8.12.3 for another example of moving database files during a restore operation.

8.12.2 Creating a Duplicate Database During a Restore Operation

You can create a duplicate copy of your database that has a unique identity from the original backed up database by using the RMU Backup and RMU Restore commands and by restoring the database, using the Duplicate qualifier. The duplicate database has the same contents as the original database but has a different and unique identity based on its creation timestamp. The default, when no Duplicate qualifier is specified in the command line or when the Noduplicate qualifier is specified in the command line, is to restore the original database and its set of files to the exact locations specified in the backup file and with the same identity as the original database that was backed up. The steps for creating a duplicate database are:

1. Create a full and complete database .rbf file.
2. Restore the full and complete .rbf file, specify the Duplicate qualifier, and specify new names and locations for the database files if they are to reside on the same system or specify new device names if the database files are going to reside on a another system with a different set of disk devices.

You can use the Directory qualifier of the RMU Restore command to move all database files (the .rdb, .rda, and .snp files) to a different directory, if desired. Use the Root, File, and Snapshots qualifiers to move the database files individually to different directories. By default, files are restored to the original device and directory locations specified in the .rbf file; this information is stored in the .rdb file that was backed up.

You can use the Directory, Root, File, and Snapshots qualifiers in combination to specify a default directory and override the default for specific files. For example, the mf_personnel database files are moved according to the qualifiers used in the RMU Restore command shown in Example 8–27.

Example 8–27 Making a Duplicate Copy of the Database and Moving Database Files to New Locations During a Restore Operation

```
$ RMU/RESTORE/DUPLICATE /DIRECTORY=DISK22:[PERS.STOR] -
_-$ /ROOT=DB_DISK2:[MFPERS] PERS_FULL.RBF -
_-$ EMPIDS_LOW/FILE=DISK44:[MFPERS]EMPIDS_LOW.RDA -
_-$ EMPIDS_MID/SHOTS=(FILE=DISK55:[MFPERS]EMPIDS_MID.SNP) -
_-$ /AFTER_JOURNAL=AIJ_DISK2:[MFPERS]MF_PERSONNEL.AIJ
$
```

In this example, the Directory qualifier causes all .rda and .snp files, except those with local qualifiers, to be moved to DISK22:[PERS.STOR]. Your original database still resides on the same system, disk devices, and directories as before you backed it up.

The following files are not located on DISK22:[PERS.STOR]:

- The .rdb file is placed on DB_DISK2:[MFPERS]MF_PERSONNEL.RDB;1. DB_DISK2 is a new logical name for \$2\$DUA88:[PERS.ROOT], its new location. The version number, ;1, indicates that because its identity is different from the original database after it is restored, this file is the only version in that directory.
- The storage area file EMPIDS_LOW.RDA;1 is moved to DISK44:[MFPERS]. DISK44 is a new logical name for \$2\$DUA44:[PERS.STOR]. The original file specification was \$2\$DUA3:[PERS.STOR.MFPERS]EMPIDS_LOW.RDA;1.
- Only the EMPIDS_MID snapshot file, empids_mid.snp;1, is located on DISK55:[MFPERS].
- A new .ajj file is created in AIJ_DISK2:[MFPERS], a new location.

You cannot use the Duplicate qualifier with an incremental restore operation in which the Incremental qualifier is specified or in a by-area restore operation in which the Area qualifier is specified. Also, the journal files cannot be interchanged between the original database and its duplicate.

8.12.3 Moving Database Files

You can move your database files, using the RMU Backup and RMU Restore commands, by restoring the database with a new name. The steps are:

1. Create a full database backup file.
2. Restore the full backup database, specifying new names or locations for the database files.

You can use the Directory qualifier of the RMU Restore command to move all database files (the .rdb, .rda, and .snp files) to a single OpenVMS directory, if desired. Use the Root, File, and Snapshots qualifiers to move database files individually to different directories. By default, files are restored to the original device and directory locations; this information is stored in the .rdb file that was backed up.

You can use the Directory, Root, File, and Snapshots qualifiers in combination to specify a default directory and override the default for specific files. For example, the mf_personnel database files are moved according to the qualifiers used in the RMU Restore command shown in Example 8–28.

Example 8–28 Moving Database Files During a Restore Operation

```
$ RMU/RESTORE/DIRECTORY=DISK2:[PERS.STOR] /ROOT=DB_DISK:[MFPERS] -
_-$ PERS_FULL.RBF -
_-$ EMPIDS_LOW/FILE=DISK4:[MFPERS]EMPIDS_LOW.RDA -
_-$ EMPIDS_MID/SHOTS=(FILE=DISK5:[MFPERS]EMPIDS_MID.SNP) -
_-$ /AFTER_JOURNAL=AIJ_DISK:[MFPERS]MF_PERSONNEL.AIJ
$
```

In this example, the Directory qualifier causes all .rda and .snp files, except those with local qualifiers, to be moved to DISK2:[PERS.STOR]. You should delete the database after backing it up and before moving it to other disk devices or directories.

The following files are not located on DISK2:[PERS.STOR]:

- The .rdb file is placed on DB_DISK:[MFPERS]MF_PERSONNEL.RDB;1 (\$2\$DUA8:[PERS.ROOT], its original location). The version number, ;1, indicates that, because you deleted the database prior to moving it, this is the only version in that directory.

- The storage area file EMPIDS_LOW.RDA;1 is moved to DISK4:[MFPERS]. DISK4 is a logical name for \$2\$DUA4:[PERS.STOR]. This file was formerly in \$2\$DUA3:[PERS.STOR.MFPERS]EMPIDS_LOW.RDA;1.
- Only the EMPIDS_MID snapshot file, empids_mid.snp;1, is moved from DISK6:[MFPERS] to DISK5:[MFPERS].
- A new .aij file is created in AIJ_DISK:[MFPERS], the existing location.

8.12.4 Moving and Updating Data Dictionary Information

OpenVMS OpenVMS
VAX Alpha

When moving your database, you can also choose to move data dictionary information. By default, the RMU Restore command uses the value of the CDD\$DEFAULT logical name to determine the data dictionary directory in which to check for the database dictionary entry. You can use the Path qualifier to specify another data dictionary directory. If the data dictionary directory that the RMU Restore command checks does not contain a copy of the database dictionary entry (or any of the metadata in the .rdb file), the RMU Restore command automatically restores the metadata information into that data dictionary directory. This means that if you specify a new data dictionary directory when you restore your database, you can create your data dictionary information in a new directory, as shown in Example 8–29.

Example 8–29 Moving Dictionary Information During a Restore Operation

```
$ RMU/RESTORE/NEW_VERSION/LOG/PATH=SYS$COMMON:[CDDPLUS]CORP_ACCT
Backup: DBS_BACKUPS:[MFPERS]PERS_FULLL.RBF
.
.
.
%RMU-I-LOGINTEGRATE, Integrating DB_DISK:[MFPERS]MF_PERSONNEL.RDB;5 into
CDD path SYS$COMMON:[CDDPLUS]CORP_ACCT
$
```

Use the Common Dictionary Operator (CDO) utility to verify that the metadata was properly integrated into the data dictionary as shown in Example 8–30.

Example 8–30 Checking That Metadata Was Moved into the Dictionary After a Restore Operation

```
$ DICTIONARY OPERATOR
CDO> DIRECTORY/TYPE=CDD$DATABASE MF_PERSONNEL
```


Use the `Nocdd_Integrate` qualifier with the `RMU Restore` command if you do not want to move your metadata information during a restore operation. Also use `Nocdd_Integrate` when you do not want to update existing dictionary information during a restore operation.

For more information about the use of Oracle Rdb databases with a data dictionary, see the *Oracle Rdb7 Guide to Database Design and Definition*, the *Using Oracle CDD/Repository on OpenVMS Systems*, and the *Oracle CDD/Repository CDO Reference Manual*.

Use the `Nocdd_Integrate` qualifier if you do not want to update data dictionary information when you restore your database, as shown in Example 8–31.

Example 8–31 Restoring a Database Without Updating the Data Dictionary

```
$ RMU/RESTORE/NOCCD_INTEGRATE/LOG PERS_FULL.RBF
.
.
.
```

When the `Nocdd_Integrate` qualifier is used, the data dictionary is not updated to record the creation of this database. The default action is to integrate the metadata information stored in the `.rdb` file into the dictionary by using one of the following methods:

- The value specified in the `Path=cdd-path` qualifier, if specified. If `cdd-path` does not start with `CDD$TOP` or `SYSS$COMMON:[CDDPLUS]`, Oracle Rdb appends the path you enter to the value of the `CDD$DEFAULT` logical name.
- If the `Path` qualifier is not specified, Oracle RMU uses the `CDD$DEFAULT` value of the user who entered the `RMU Restore` command, and the database file name.

See Section 8.12.4 for specific examples about moving dictionary information. Oracle RMU ignores the `Path` qualifier if the user also specified the `Nocdd_Integrate` qualifier.

The `Nocdd_Integrate` qualifier is useful because it can prevent the `RMU Restore` command from copying metadata information into the data dictionary from the `.rdb` file, if data dictionary information does not exist in the default or specified data dictionary directory. Thus, the database restore operation can occur, and then links with the data dictionary can be established afterwards.

Note

If the CDD_INTEGRATE action fails, the restored database is still available because the database has been restored successfully, assuming no restore error messages were received. Only the metadata was not successfully copied into the data dictionary. You can correct the problem and then use either the SQL INTEGRATE DATABASE statement specifying the PATHNAME . . . ALTER DICTIONARY arguments if the dictionary entity already exists, or the FILENAME . . . CREATE PATHNAME arguments if the dictionary entity does not exist. See the *Oracle Rdb7 SQL Reference Manual* for more information.



8.13 Using the SQL EXPORT and IMPORT Statements

Use the SQL EXPORT and IMPORT statements to do the following:

- Reorganize an Oracle Rdb multifile database.
- Reorganize an earlier version (pre-V3.0) of an Oracle Rdb single-file database (or one defined using Oracle Rdb without specifying at least one SQL CREATE STORAGE AREA clause) into a multifile database with at least one .rda file.
- Reload an Oracle Rdb database, leaving the storage area definitions the same but changing the device specifications.
- Make a copy of an Oracle Rdb database for archival purposes.

The SQL EXPORT and IMPORT statements are not intended for regular backup operations of a database. For regular backup and restore operations with Oracle Rdb databases, use the RMU Backup and RMU Restore commands.

For more information on the SQL EXPORT and IMPORT statements, see the *Oracle Rdb7 SQL Reference Manual* and the *Oracle Rdb7 Guide to Database Design and Definition*.

After-Image Journaling and Recovery

Oracle Rdb provides two types of journaling:

- Before-image journaling, also known as recovery-unit journaling—Records each data modification *before* it is committed to the database.

Oracle Rdb creates a recovery-unit journal (.ruj) file automatically for each process that performs an update transaction. If a user's transaction terminates abnormally or if the user invokes an SQL ROLLBACK statement, Oracle Rdb *rolls back* the transactions that are logged in the recovery-unit journal file.

- After-image journaling—Records each data modification *after* it is committed to the database.

This optional journaling method requires that you explicitly enable after-image journaling and add after-image journal (.aij) files for the purpose of recording after-images of each update transaction committed to the database. If a failure occurs, Oracle Rdb rolls forward the transactions that are logged in the after-image journal files.

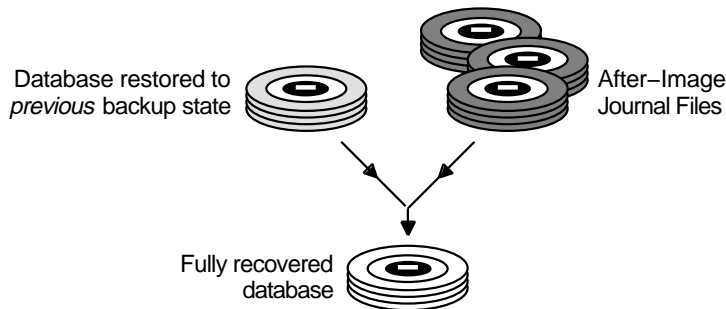
This chapter describes how to use after-image journaling in conjunction with regular database backup operations as a method of maintaining data integrity.

9.1 Introduction

After-image journaling protects the integrity of your data by recording all database transaction activity to an after-image journal (.aij) file. Oracle Corporation recommends you enable after-image journaling to record your database transaction activity between full backup operations as part of your database restore and recovery strategy.

Figure 9–1 shows how you use the transactions recorded in after-image journal files along with Oracle Rdb backup files to restore a database to a specific point in time.

Figure 9–1 Restoring Data from the After-Image Journal Files



NU-3611A-RA

Because after-image journaling provides a method to roll forward all transactions since the last backup operation, it is possible to recover an entire database or just a single database page up to the last successfully committed transaction.

9.1.1 Recommended and Required Usage

After-image journaling is optional except when you use the FAST COMMIT or the COMMIT TO JOURNAL optimizations (specified using the SQL ALTER DATABASE statement). However, Oracle Corporation recommends that you always enable after-image journaling to record your database transaction activity. Journaling typically costs 5 to 10 percent in overhead, but provides a continuous backup of database transactions.

To be effective, you should enable after-image journaling for the entire time that the database is open and active. Do not sporadically enable and disable after-image journaling. For example, do not disable after-image journaling during batch processing, because this leaves gaps in the data modifications recorded in the journal file and defeats the purpose of using journaling. A database that does not have after-image journaling enabled is not recoverable.

Recommended Usage

When after-image journaling is enabled, any process that is attached to the database (including the database recovery (DBR) process) writes to the after-image journal file. Oracle Corporation recommends that you enable after-image journaling at all times, but journaling is especially important when:

- You cannot easily apply all transactions to the database that have occurred since the last full backup operation
- You back up and restore a single database storage area

See Chapter 7 for more information on using backup strategies by storage area.

Required Usage

You must enable after-image journaling when you use either of these options:

- FAST COMMIT optimization
- COMMIT TO JOURNAL optimization

Your overall database system availability and performance can improve when your application takes advantage of these performance optimizations along with after-image journaling.

See Section 9.8 for more information about the fast commit and the commit-to-journal optimizations.

9.1.2 Information Written to the After-Image Journal File

The information that is journaled includes:

- New or updated data written to the database root (.rdb) file (including new metadata and metadata changes)
- New or updated data written to storage area (.rda) files
- The following records:
 - AIJ header
 - ACE header
 - Checkpoint
 - Control (commit or rollback)
 - Option information
 - Close
 - Information
 - Shuffle
 - Prepare

Information that is not journaled includes the following modifications that you make with the SQL ALTER DATABASE statement:

- Changing the number of database users
- Changing the number of cluster nodes
- Reserving journal files
- Suppressing and unsuppressing journal files for multiple fixed-sized journal files
- Reserving storage area files

- Moving storage areas to different disks
- Moving the database root file to another disk

These changes are not journaled because they are not executed in a read/write transaction and cannot be rolled back.

9.1.3 Journaling Strategy

As you devise and implement a sound journaling strategy for your database application, the decisions you make should be based on careful consideration of the following additional factors:

- Journaling: to use a single extensible journal file versus multiple fixed-size journal files
- Location of the after-image journal files
- AIJ backup operation:
 - To automate or semiautomate AIJ backup operations
 - To back up files prior to each database full backup operation versus performing journal file backup operations as each backup file becomes full
 - To perform noquiet-point AIJ backup or quiet-point AIJ backup operations
 - To magnetic tape directly, or first to disk and then to magnetic tape
 - To back up journal files sequentially as each journal file becomes full, or backup several journal files simultaneously by specifying unique journal sequence numbers (using the Sequence qualifier)
- AIJ files: to optimize or not to optimize backup files
- Disk resources: effects of limited disk resources
- WORM areas: to journal or not to journal write-once areas on write-once, read-many (WORM) media
- Database performance: to optimize database performance, for example, by enabling fast commit transaction processing versus accepting normal performance
- Database availability: to require high versus moderate database availability
- Transaction activity: to support high versus normal or low transaction activity

- Recovery management: to require less versus more database administrator (DBA) file management intervention

This chapter addresses many of these considerations.

9.1.4 Displaying After-Image Journaling Performance

To find out what effects after-image journaling has on performance, use the RMU Show Statistics command to invoke the performance monitor. This command monitors database disk I/O operations and the number of completed database transactions.

For a description of the performance monitor and the RMU Show Statistics command, see the *Oracle Rdb7 Guide to Database Performance and Tuning*.

9.2 Enabling After-Image Journaling

Use the following SQL statements or Oracle RMU commands to enable after-image journaling:

- SQL CREATE DATABASE statement
- SQL ALTER DATABASE statement
- RMU Set After_Journal command
- RMU Restore Aij_Options command (if you are restoring a database)

Examples 9–1 and 9–2 provide brief SQL and Oracle RMU examples that enable after-image journaling using multiple journal files.

Example 9–1 Enabling After-Image Journaling Using SQL Statements

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> RESERVE 4 JOURNALS❶
cont> ADD JOURNAL JOURN_1 FILENAME USER4:[JOURNAL]JOURN_1.AIJ❷
cont> ALLOCATION IS 1024 BLOCKS❸
cont> EXTENT IS 512 BLOCKS❹
cont> ADD JOURNAL JOURN_2 FILENAME USER4:[JOURNAL]JOURN_2.AIJ
cont> ALLOCATION IS 1024 BLOCKS
cont> ADD JOURNAL JOURN_3 FILENAME USER4:[JOURNAL]JOURN_3.AIJ
cont> ALLOCATION IS 1024 BLOCKS
cont> JOURNAL IS ENABLED;❺
%RDMS-W-DOFULLBCK, full database backup should be done to ensure
future recovery❻
cont> EXIT;
$
$ RMU/BACKUP MF_PERSONNEL MFPERS_53196.RBF
```

❶ Determine how many journal slots you need and reserve them. By default, one journal slot is reserved in the database root file.

❷ Add journal files.

By adding a second journal file, you automatically override the default journaling method (a single extensible journal file) and invoke multiple fixed-size journal files. For availability, you should locate journal files on physical devices separate from the devices holding the database files.

Journal file names must be unique.

❸ To improve performance, allocate an initial amount of disk space to the after-image journal file. The default value is 512 blocks. (See Section 9.4.3 for information about setting the allocation size.),

❹ If you are using a single extensible journal file, specify the size of after-image journal file extents. The default value is 512 blocks.

Note: The extent value is ignored if you are using fixed-size journal files. (See Section 9.4.4 for more information on the relationship between file allocation and extent size for an extensible journal file.)

❺ After you add all the journal files, enable journaling. You can also enable journaling using the Enable qualifier on the RMU Set After_Journal command.

Because reserving journal slots is an operation that is not journaled, a warning message displays to indicate that you should do a full backup operation of your database to ensure recovery.

- ⑥ A warning message displays indicating that a full database backup operation should be done to ensure future recovery when you reserve journal slots. When you reserve journal slots to create additional journal files, this operation is not journaled and therefore requires a full database backup operation to ensure database consistency.

See Section 9.6 for more information about performing backup operations on after-image journal files.

Example 9–2 shows how to enable after-image journaling using Oracle RMU commands on a Digital UNIX system.

Example 9–2 Enabling After-Image Journaling Using RMU

```
$ rmu -set after_journal \  
> -reserve=4 \  
> -add=\(name=JOURN_1, file=/usr/journal/journ_1, \allocation=1024) \  
> -add=\(name=JOURN_2, file=/usr/journal/journ_2, \allocation=1024) \  
> -add=\(name=JOURN_3, file=/usr/journal/journ_3, \allocation=1024) \  
> -enable \  
> mf_personnel  
%RMU-W-DOFULLBCK, full database backup should be done to ensure future  
recovery  
$  
$ rmu -backup mf_personnel mfpers_53196.rbf
```

Example 9–1 and Example 9–2 provide a starting point for the discussions in this chapter. Refer to the *Oracle Rdb7 SQL Reference Manual* and the *Oracle RMU Reference Manual* for complete command syntax and usage information, and for more examples.

Note

When you enable after-image journaling on OpenVMS systems, the journaling operation impacts the size of the global section. After-image journaling uses up to 1200 additional OpenVMS pages in memory (each 512 bytes) to store AIJ request blocks.

As a process modifies a data record, the after-image of the record is copied into one or more AIJ request blocks, and then the AIJ request block is queued for submission into the after-image journal file. Each AIJ request block is approximately 2000 bytes in length. Thus, to modify a 3000-byte data record, the journaling operation requires two AIJ request blocks.

9.3 Disabling After-Image Journaling

You can use SQL statements or Oracle RMU commands to disable after-image journaling. However, Oracle Corporation recommends that you keep after-image journaling enabled at all times.

There are some database operations that disable journaling automatically. The following list describes these operations and the actions taken by Oracle Rdb in response:

- When you export and subsequently import a database that has after-image journaling enabled, after-image journaling is automatically disabled. Therefore, you must manually re-enable after-image journaling after you import a database.
- If you perform a convert operation (RMU Convert) when after-image journaling is enabled, Oracle RMU disables after-image journaling during the database conversion and then reactivates journaling.
- If you perform a convert operation and include the Rollback qualifier on the RMU Convert command, Oracle RMU disables after-image journaling and returns the message: RMU-I-CANTENAAIJ. Oracle Corporation recommends that you enable after-image journaling when the RMU Convert command and the rollback operation complete.

Refer to the *Oracle Rdb7 Installation and Configuration Guide* and the *Oracle RMU Reference Manual* for complete information about handling after-image journaling when converting databases.

9.4 The After-Image Journal (.aij) File

When you enable after-image journaling, it saves copies of the modified rows (plus additional information) to a journal file. The default file type for an after-image journal file is .aij.

Consider these points as you set up your after-image journal files:

- You should designate that the after-image journal files be written to disk and the disk should not contain any other database files. You should restrict the contents of the disk to hold only after-image journal files. Otherwise, in case of hardware problems, you might not be able to recover the database.
- On OpenVMS cluster systems, ensure that the after-image journal disk is accessible from all cluster nodes at all times. Ideally, the after-image journal disk should be served by multiple controllers.

- You can choose to use either one extensible journal file or multiple fixed-size after-image journal files.
- You can enhance performance by understanding how the allocation and extent sizes affect the I/O operational costs.

Sections 9.4.2 through 9.4.4 describe after-image journal file considerations in more detail.

9.4.1 Location and Accessibility

When after-image journaling is enabled, Oracle Rdb automatically logs all data modifications (except those mentioned in Section 9.1.2) to the after-image journal file and to the database file. To maintain high availability and performance, after-image journal files should always reside on disks that are *separate* from the database root (.rdb) file and storage area (.rda) files, snapshot (.snp) files, and recovery-unit journal (.ruj) files.

If possible, you should also locate each fixed-size journal file on a separate device. Locating each fixed-size journal file on a separate disk minimizes risks associated with after-image journal file loss or unavailability should a device fail or be brought off line. For example, if two or more after-image journal files reside on the same failed device, the loss of information or its unavailability is far greater than that of a single after-image journal file.

Using Logical Names (OpenVMS Only)

OpenVMS OpenVMS
VAX Alpha

You can define one or more system logical names to refer to the after-image journal file. The default directory for the journal file is the same as for the database files. Include a full file specification to specify each device and directory.

Do not specify a version number to ensure that the most current version of the file is used. The file specification can be a system logical name. For example, the sequence of commands shown in Example 9-3 defines two system logical names that refer to the device and directory of two after-image journal files.

Example 9–3 Placement of the After-Image Journal File

```
$ DEFINE/SYSTEM PERS$JOUR1 "DISK1$:[DATABASE.JOURNAL]JOURN_1.AIJ"
$ DEFINE/SYSTEM PERS$JOUR2 "DISK3$:[DATABASE.JOURNAL]JOURN_2.AIJ"
$
$ SQL
SQL> ALTER DATABASE
cont> FILENAME DISK2$:[DEPT3]MF_PERSONNEL
cont> ADD JOURNAL JOURN_1 FILENAME 'PERS$JOUR1'
cont> ALLOCATION IS 1024 BLOCKS;
cont> ADD JOURNAL JOURN_2 FILENAME 'PERS$JOUR2'
cont> ALLOCATION IS 1024 BLOCKS;
SQL> EXIT
```

Cluster Accessible Disks

You can keep the after-image journal files on public disks or private disks, but, if your database is configured for access in a cluster, you must ensure that all after-image journal files are accessible by all nodes in the cluster. Therefore, you should place the files on dual-pathed disks that are accessible to all nodes in the cluster. This allows for alternative access paths in the event that one path fails and ensures that the files are accessible by all nodes in the cluster.

See the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information about placement of Oracle Rdb database files in a cluster environment.

9.4.2 Extensible and Fixed-Size Journal Files

You can choose to use one or multiple after-image journal files:

- One extensible journal file
When you enable after-image journaling, a single extensible journal file is created by default. When the journal file's allocated space is filled, the file automatically extends itself to accommodate more data.
- Multiple fixed-size journal files
Uses three or more fixed-size journal files, one at a time, in a circular fashion. When a file's allocated space is filled, journaling switches to a second fixed-size journal file until it becomes full, and so forth. The switch to a new journal file occurs automatically and transparently, as though the multiple journal files are one seamless journal file. A single transaction can span one or more journal files.

Oracle Corporation recommends that you use multiple fixed-size after-image journal files for both single-file and multifile databases. With this method, after-image journal file management and failure recovery is automatic; operator intervention is not required under most circumstances.

Note

You must have a minimum of two journal files available (three are recommended) at all times or the journaling system changes over to extensible journaling.

When you use multiple fixed-size journal files, all journal files are readily available on separate disks to provide high availability for the database.

Although the default is to use only one extensible journal file, a single extensible journal file can impose a number of disadvantages to the performance and recoverability of your database. Table 9–1 compares the two methods of after-image journaling.

Table 9–1 Comparison of Single Versus Multiple After-Image Journal Files

Category	Disadvantages of Using Single Extensible File	Advantages of Using Multiple Fixed-Size Files
Backup operations	For applications with high transaction activity, it may be nearly impossible to back up a journal file without interrupting database activity. That is, committed transactions could be written to the journal file faster than the file can be backed up, eventually leading to periods of interrupted database activity while the AIJ backup operation completes.	High performance AIJ backup capability allows you to immediately back up journal files, or automate after-image journal backup operations. For example, you can write a DCL command procedure on OpenVMS systems, or enable the AIJ backup server (ABS) to back up any journal file as soon as it fills up.
Disk space	Because you cannot limit the number of times the journal file extends itself, it is possible that without closely monitoring disk space, you can run out of space on the after-image journal device for applications that have moderately high transaction activity.	Journal files fill up and automatically switch over to the next available journal file. User processes automatically perform a checkpoint operation after switching journal files. This avoids the problem of having a long checkpoint frequency or a long transaction span all available journal files, which could lead to a condition similar to running out of disk space.

(continued on next page)

Table 9–1 (Cont.) Comparison of Single Versus Multiple After-Image Journal Files

Category	Disadvantages of Using Single Extensible File	Advantages of Using Multiple Fixed-Size Files
Fast commit processing	You need twice the disk space to back up the journal file. The backup operation creates a second journal file on the same disk as the existing journal file. You must closely monitor available disk space to ensure that there is space for the backup operation to complete. If you find that there is insufficient disk space during a backup operation, you may need to close the database or disable fast commit transaction processing temporarily to allow the backup operation to complete.	Provides undo/redo (fast commit transaction processing) performance without after-image journal file backup or media recovery concerns.
Recovery	After-image journal file management and failure recovery is considered difficult, labor-intensive, and time-consuming, and often requires exclusive database access in order to resolve journal file management issues.	Automatic recovery from failures occurs when a journal file becomes unavailable for some reason (such as a disk device being accidentally shut down). Because all journal files are available on disk, you can restore and recover the database, an area, or a page using only one RMU Restore command.
Performance	If you specify a large value for the SQL EXTENT IS clause, the application can be <i>stopped for several minutes</i> while the journal file extends and is initialized.	Does not require journal file initialization at run time.

See Section 9.7 that describes after-image journal file switchover in more detail.

9.4.3 Setting the Allocation Size

You can specify the number of blocks initially allocated to the after-image journal file when you enable after-image journaling. The allocation setting controls the physical size of the file created for after-image journaling. The allocation size also determines the physical end-of-file (PEOF). The default allocation value is 512 blocks. (See Section 9.2 for SQL and Oracle RMU examples that show how to set allocation sizes.)

For example, if you set the after-image journal file allocation to 5000 blocks and set the file extent to 1000 blocks, the file size is set to 5000 blocks (plus or minus a few blocks if rounding is required).

To determine an appropriate allocation setting, estimate the amount of data that is written to the after-image journal file in one day. Use the RMU Show Statistics command to establish an approximation for blocks per transaction

(BPT) and transactions per second (TPS). For example, assuming an 8-hour workday, calculate the amount of data for one day as follows:

$$\text{Amount of data} = 8 \text{ hours} * 3600 \text{ seconds} * BPT * TPS$$

Set the allocation value to 25 percent higher than one day's worth of data. In addition, use the following recommendations for extensible and fixed-size journal files.

For a Single Extensible Journal File

The allocation size you set for a single extensible journal file depends on how frequently you back up the journal file. The backup journal file can be subsequently backed up to tape, reducing the amount of disk space required.

Because performance typically degrades when an after-image journal file is extended, you should set the initial allocation size to result in the least possibility for extension. The following table provides guidelines for setting allocation sizes:

For Databases with . . .	Then . . .
High transaction volumes	Set a larger allocation size to reduce journal file extension and improve performance significantly.
A large number of updates and mission-critical response-time requirements	Preallocate the journal file to prevent it from being extended during normal processing. For example, specify an allocation size for the file that is large enough to handle processing for one day. Then, back up the after-journal file to tape once a day.
One disk device dedicated to the after-image journal file	Choose an allocation size that takes up the entire disk.
Fast commit transaction processing enabled	Allocate twice the file space on the after-image journal disk if you have fast commit transaction processing enabled and you plan to back up the after-image journal file. You need sufficient file space because the backup process must compress and retain some fraction of the original after-image journal file or files. This fraction may approach 100 percent of the original size of the journal file. Therefore, be sure to reserve enough disk space to duplicate the maximum size of the file. See Section 9.8.1.3 for more information.

See Section 9.4.4 for more information on the relationship between file allocation and extent sizes.

For Multiple Fixed-Size Journal Files

For multiple fixed-size journal files, specify an allocation size for the after-image journal file based on the previous math equation that calculates the amount of data. You can specify a different allocation for each journal file. For example, file allocation size might differ depending on disk device capacity.

Also, if you enable the AIJ backup server (ABS), each full after-image journal file is backed up automatically after journaling switches to writing to the next available journal file.

9.4.4 Setting the Extent Size for an Extensible Journal File

You use the SQL EXTENT IS parameter to specify how many blocks Oracle Rdb uses to extend the journal file. When you use a single extensible journal file, you should understand costs involved when you extend the journal file versus preallocating a large number of blocks for an extensible file.

The after-image journal file extent serves two purposes:

- Controls how many blocks the file extends if it grows beyond the original allocation size
- Indicates the number of blocks to use for initializing the after-image journal file, and serves as the logical end-of-file (LEOF)

For example, whenever the LEOF is updated, all the blocks in the new range are also initialized with -1 values. So, if a file allocation is 5000 blocks and the file extent is 1000 blocks, on first access the first 1000 blocks are filled with -1 values and the after-image journal file is created as 1000/5001. The LEOF is 1000, the PEOF is 5001, and blocks 1 to 1000 are filled with -1 values. Once the after-image journal file fills up to virtual block number (VBN) 1000, the LEOF is extended to 2000 blocks, and blocks 1001 to 2000 are filled with -1 values.

The following sections provide information to help you understand journal file initialization and locking, and how to set the SQL EXTENT IS parameter to reduce the number and size of after-image journal file extensions.

Estimating Cost Factors

A process needs to determine the location of the logical end-of-file. To do this, each user process holds a special lock, called the after-image journal lock. The lock-value block maintains information regarding the location of the end of the after-image journal file. If the lock-value block does not have information, the lock-value block is rebuilt.

To estimate the cost of rebuilding the after-image journal lock and extending the size of the journal file, use the following calculations:

- Rebuilding the after-image journal lock

To estimate the cost of rebuilding the after-image journal lock, use the following formula to find the number of I/O operations required to establish the logical end-of-file (LEOF) pointer:

$$(\ln \text{ AIJ size} / \ln 2) + \text{number of nodes} =$$

Number of I/O operations to establish the LEOF

The notation “ln” refers to the natural log scientific notation.

For example, for a database with 16 nodes and an extent size set to 1000 blocks, the number of I/O operations needed to find the end of file is calculated as follows:

$$LEOF = (\ln 1000 / \ln 2) + 16 = 26$$

- Extending the extensible after-image journal file

In general, you can improve performance by increasing the number of blocks for file allocation to reduce the number of times that Oracle Rdb must extend an extensible journal file. This is particularly true for applications with high transaction volumes. The larger the extent size, the longer it takes Oracle Rdb to rebuild the lock.

An after-image journal file extends at the rate of approximately 2000 blocks (512 bytes) per second (1Mb per second). While the file is extending, the database is stalled such that transactions are not allowed to commit or roll back.

For example, assume that a journal file has been allocated with 100,000 blocks and that about 2000 blocks can be initialized per second. The following table provides examples to demonstrate how the extent size affects the amount of time it takes to fill the extent section of a 100,000 block file.

If the extent size is	Then . . .
. . .	
100,000 blocks	The 50-second cost is incurred all at one time.

If the extent size is ...	Then . . .
10,000 blocks	The after-image journal file is extended 10 times, but each extent and initialization takes about 5 seconds.

You need to evaluate these cost estimates to determine how to incur this I/O operational expense with the least effect to your business applications. For example, you must decide if it is less disruptive to have the first user wait 1 second to rebuild the after-image journal lock (based on an extent size of 100,000 blocks). Compare this to the expected disruption of 0.1 seconds each time the file is extended (based on the smaller extent size).

As another example, consider the following options and analysis for allocating and initializing a 200,000-block after-image journal file:

- You can specify an extent size of 200,000 blocks.
The drawback to this method is that the first user process to attach to the database stalls for some time and all transactions are prevented from writing to the after-image journal file while the file is initialized. You can work around this problem by opening the database on a node so you can avoid the lock rebuild of the first user. In addition, lock rebuilds are slow. However, if you specify a large extent size, you need to rebuild an AIJ lock only when a process holding the lock aborts.
- You can specify an extent size of 10,000 blocks.
This means the file is extended 20 times for a 200,000-block allocation and should not produce a significant stall. Furthermore, lock rebuilds proceed much faster.

9.5 Journaling List (WORM) Data

Because list data stored in write-once storage areas on write-once read-many (WORM) optical media usually consists of large objects resulting in large after images when journaling is enabled, you should consider carefully whether or not you need to journal WORM list data.

For example, if many lists are stored, such as during large data-load operations, it could result in:

- Large after-images quickly filling a journal file. Thus, you must carefully plan disk and tape resources. (When after-image journaling is enabled, journal files are written to a disk device and after-image journal file backup operations written to tape media.)

- Less than optimal performance for insert and update operations.

The decision to journal list data on WORM media is based on your application, your recovery strategy in the event of a WORM media failure, the volume of list data to be stored, the cost of managing and maintaining a journal file, performance considerations, and finally weighing alternative strategies.

You should select a strategy that works best for your application, given the risks, and that ensures the desired outcome should you experience a WORM media failure.

The following discussions take several factors into consideration and weigh them against a variety of best-case to worst-case scenarios to determine the risk factors for temporary and permanent loss of data, loss of space on WORM media, and the costs associated with these potential losses.

Reloading Instead of Journaling

In most cases, databases on WORM devices that include stable, read-only data do not require that you use after-image journaling. Such list data may be updated monthly, quarterly, or annually and can be done as a data-load or a by-area restore operation. Because this is essentially a batch load operation, journaling these transactions is unnecessary if the load operation is immediately followed by a database backup operation.

In this case, do not journal the data-load operation. Recovery from media loss involves media replacement with list data being reloaded through a data-load or a restore operation. The benefit of this strategy is that you do not need to perform backup, restore, or recover operations of list data on WORM media. To use this strategy, exclude the write-once area on the WORM media from your backup operations by including the `Exclude=storage-area-name` qualifier on the RMU Backup command.

To disable journaling for a write-once storage area on WORM media, use the `WRITE ONCE (JOURNAL IS DISABLED)` clause of the SQL `CREATE DATABASE` or `SQL ALTER DATABASE` statement.

Performing Backup Operations Only

Consider the case of a write-once area on WORM media that is backed up on a regular basis but is not being journaled.

If there is WORM media failure, the information in the most recent backup file is restored to new WORM media. However, because there is no after-image journal file, all information written since the time of the last backup operation is unrecoverable. In this case, the information is lost and the area is inconsistent.

To work around this problem, you must use the `Worm_Segments` qualifier with the `RMU Repair` command to set columns to null if the columns contain damaged or missing list segments. Performing a repair operation prevents the database from raising exceptions and closing down if there is a fetch of information from any of the missing pages in the area.

If you disable journaling for write-once areas on WORM media, you should understand the consequences of WORM media failures and take all precautions to prevent data loss under all circumstances.

One possible alternative is to mirror or shadow the database root file and the WORM media. This option is based on the assumption that it is important to have list data accessible and available at all times. This approach does not require that either the database root file or list data ever be restored or recovered.

Performing After-Image Journaling Only

At the other end of the spectrum are applications that continually insert and update list data on WORM media. A strategy to journal all committed list data transactions requires that you plan accordingly. The insert and update performance with this strategy may be less than optimal.

For example, you must ensure that journal files are always available and when a file becomes full you must back it up immediately so the journal file becomes available again. To maintain data consistency, it is essential that you include the WORM media as a part of your routine during backup, restore, and recovery procedures.

Furthermore, recovering from lost WORM media requires that you restore and recover the information to another WORM media if the original media cannot be placed back on line. Because you cannot write over what is already written on WORM media, allocate sufficient space so that you can restore and recover the list data.

In addition, if the database root file needs to be restored due to a disk failure, you are likely to waste additional space on WORM media even though information on the WORM media is unaffected. This is because any preallocated unused space in the write-once storage area space between the highest page number to the end of the file allocation or extent is lost after you restore and recover the database root file and preallocate more storage space. On average, the loss of space is usually half the extent size.

9.6 Backing Up After-image Journal Files

After-image journaling is completely integrated with the Oracle RMU backup operation to enable the database to be restored to a specific point. The database backup operation contains all changes to pages, and the after-image journal backup file contains all changes to rows and index records that have occurred since the last backup operation. Thus, all update transactions that were active when the backup operation started or that began after the backup operation started must be recovered from a backup copy of the after-image journal file.

Use either of the following backup methods to record the contents of the after-image journal file to a journal backup file that can be used to recover a database:

- Automatically using the AIJ Backup Server (ABS)
- Manually using the RMU Backup After_Journal command

You should perform backup operations during quiet points (by using the Quiet_Point qualifier) to maintain a transaction-consistent view of the database.

Note

Do not use other backup utilities such as the OpenVMS Backup (BACKUP) utility or the Digital UNIX `tar` function to back up and restore Oracle Rdb databases or after-image journal files. Relying on backup utilities other than Oracle RMU can compromise the reliability and availability of the database. See Section 7.4 for more information.

The following sections describe after-image journal backup operations.

9.6.1 Backup and Recovery Strategy

When devising an after-image journal file backup strategy, consider the following options for recording and recovering database changes:

- You can accumulate multiple after-image journal files (the current journal file plus all backed-up journal files).
- You can perform on regular incremental backup operations of the database between full backup operations.

The best strategy depends on the time between full backup operations compared to the time and effort to restore the database. The longer the time between full backup operations, the greater the advantage of using the incremental backup operation to keep a record of database changes. If you perform a full backup operation every few days, you might benefit most by creating one or more after-image journal files, especially if the files are not too large.

For example, consider a scenario in which you accumulate 2 weeks' worth of daily after-image journal files since your last full backup operation. You must restore the entire database and recover each of the 14 journal files in proper sequence, one at a time, to bring your database to its most current state.

This procedure may take much longer to accomplish than restoring the full backup file and incrementally restoring the latest incremental backup file to bring the database to its most current state. This is because each night's incremental backup operation accumulates each successive day's changes into just one incremental backup file.

9.6.2 Disk and Tape Backup Media

The `RMU Backup After_Journal` command provides a two-stage journaling process that can save disk space and reduce the dependence on tape drives. The `RMU Backup After_Journal` command copies completed transactions recorded in the primary journal file (always on a disk device) to the backup file (which may be on a tape device or disk device). Oracle Rdb recommends you back up journal files to disk, then perform another backup operation to back up these files (from the backup disk) to tape.

Note

Do not back up large extensible journal files directly to tape.

You can back up the after-image journal file on line, while users are attached to the database. You can back up your journal file either all at once or continuously.

By default, the `RMU Backup After_Journal` command backs up the entire contents of your journal file to tape. If there are no after-image records in your journal file, Oracle RMU returns the following error message:

```
%RMU-I-EMPTYAIJ, after-image journal file is empty
```

If your backup operation results in a large number of backup tapes, consider purchasing software such as a storage library system or archive backup system to manage the tapes.

9.6.3 Reusing Disk and Tape Backup Media

To determine when you can reuse after-image backup media, follow this procedure:

1. Dump the database root file information and the after-image journal file information using the RMU Dump Header command.
2. Examine the dump header output for the following information:

- Transaction sequence numbers

Find the after-image journal sequence number in the after-image journal file dump output and compare it with the after-image journal sequence number in the database root file. For example:

```
$ rmu -dump -after_journal mfpers.aij
```

```
.
.
.
  AIJ Sequence Number is 0
```

```
.
.
.
$ rmu -dump -header mf_personnel
```

```
.
.
.
  AIJ Sequence Number is 1
```

- Inconsistent storage areas. For example:

```
Storage area EMP_INFO
.
.
.
Status...
  - Area is marked inconsistent
    Consistent to TSN 75
    Roll-forward sequence number is 0
```

3. Reuse the backup media according to the guidelines in the following table:

If There Are . . .	Then Reuse the Backup Media . . .
No inconsistent storage areas	If the sequence number of the after-image journal file is less than the sequence number stored in the database root file.
Inconsistent storage areas	If the sequence number of the after-image journal file is less than the sequence number stored in the storage areas.

9.6.4 Scheduling

For an extensible journal file, you must change your journal file:

- Before the database full backup operation
You must enable after-image journal file backup operations before the full backup operation to ensure a recoverable database. If you need to recover your database, rollforward operations commence with the active after-image journal file, thus increasing the speed with which rollforward operations complete.
- When there is a lack of available resources
This is particularly important if you use an extensible journal. If your extensible journal file becomes low on disk space, perform an after-image journal file backup operation. The backup operation truncates the extensible journal file to its initial allocation, thus making additional disk space available. If you continue to have space problems on the disk where the journal file resides, consider moving the journal file to another disk.

For fixed-size journal files, all journal files (except for the current active journal file) should be backed up. When the backup operation starts, a switch to the next available journal file occurs automatically. Journaling operations continue on a new journal file so that backup operations can proceed on the now unavailable journal file. Section 9.6.5 provides information about switching journal files manually.

Note

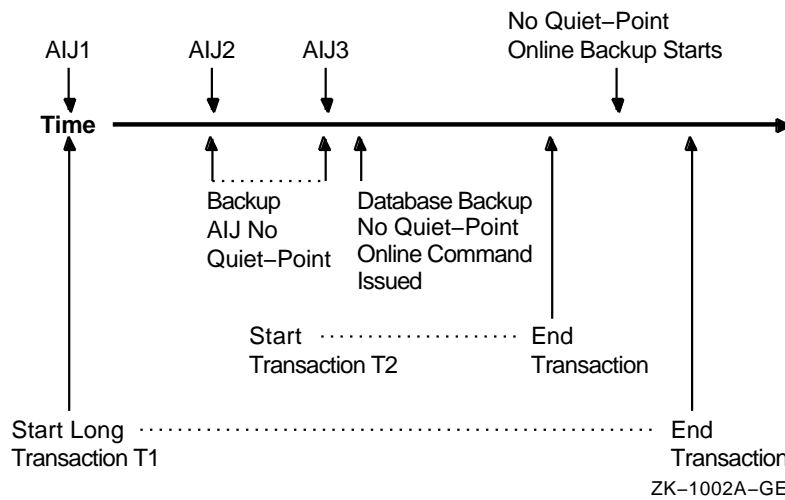
Always perform the database backup operation on line to allow database activity to continue during the backup procedure.

9.6.5 Transaction Sequence Numbers (TSN)

Because the purpose of performing after-image journal file backup operations is so you can recover all update transactions in the event of a failure, the backup operation must record all transactions that are active when the backup operation starts or that begin after the backup operation starts.

Oracle Rdb uses transaction sequence numbers to ensure the correct recovery of all transactions. Because you back up the journal file before the database backup operation begins, consider the possibility of having the same transaction recorded in both the after-image journal backup file and the database backup file, as shown in Figure 9-2:

Figure 9-2 When to Back Up the After-Image Journal File



Oracle Rdb can handle these situations because it assigns a transaction sequence number (TSN) to each transaction as it is committed. When Oracle RMU performs a database backup operation, it saves the last recorded TSN in the database root file. Then, if you need to restore the database, the database root file is updated with the last or highest committed TSN for the database.

When you recover a journal file, Oracle Rdb compares the highest committed TSN in the database root file with the TSN written in the open record of the journal file. If the TSN values are the same, the recovery operation proceeds. Thus, the backup, restore, and recovery operations take the correct action based on comparisons of TSN values. For example:

If . . .	Then . . .
<p>You perform a quiet-point after-image journal file backup operation and a quiet-point database backup operation</p>	<p>The backup operations proceed only when there are no active transactions. In Figure 9–2, the backup operation does not start until after transaction T1 completes. The restore operation returns informational messages that indicate recovery must start with the TSN numbers written to the AIJ3.AIJ file (shown in Figure 9–2).</p> <p>For fixed-size journal files, you can use the <code>Switch_Journal</code> qualifier with the <code>RMU Set After_Journal</code> command to manually switch the current journal file to another available empty journal file before backing up the database and journal files. By manually switching journal files, you eliminate the need for the recovery operation to examine transactions that have already committed before the backup operation starts if you must later restore and recover the database.</p>
<p>You perform a noquiet-point after-image journal file backup operation and a noquiet-point database backup operation</p>	<p>The backup operation determines which transaction you must recover first and a restore operation returns informational messages to indicate that recovery must start with the after-image journal sequence number that represents the oldest active write transaction. For example, because of the long running transaction in Figure 9–2, a restore operation requires the AIJ1 journal file (transaction T1) is recovered first to start recovery after the database restore operation.</p> <p>When you use multiple fixed-size files, a journal file switchover can occur at any time and Oracle Rdb handles the switchover correctly. In most cases, there is no benefit to manually switching journal files because active transactions span journal files. Performing a journal file switchover results in the active transactions spanning an additional journal file. Thus, performing a journal file switchover on a partially full journal requires that you recover an extra journal file if you must restore and recover your database. See Section 9.6.8.1 for more information.</p>

Note

Do not use the `Overwrite` qualifier (on the `RMU Set After_Journal` command) to perform write I/O operations to journal files before you back up those files. Oracle Corporation does not recommend writing over journal files because it makes the database nonrecoverable. Use the `Overwrite` qualifier only for databases that you can reconstruct without using journal files.

9.6.6 Backup Termination or Failure

When an after-image journal file backup operation terminates prematurely, the database root file is initialized to reflect the termination. If this happens, you must rerun the after-image journal file backup utility to completion.

Note

Do not discard the backup file produced by the failed after-image journal file backup operation. Discarding the file might lose information required to recover the database.

9.6.7 Backing Up a Single Extensible Journal File

You can back up your extensible journal file either all at once or continuously. By default, the `RMU Backup After_Journal` command backs up the entire contents of your journal file to tape. However, if the journal file is large, Oracle Corporation recommends that you back up directly to disk.

If you are using an extensible journaling system, you should carefully monitor the size of the journal file, particularly during periods of heavy use. When the size of the extensible file exceeds the disk space allocation, all database activity can stop.

See the *Oracle RMU Reference Manual* for information about using the `RMU Backup After_Journal` command to back up extensible journal files.

9.6.8 Backing Up Multiple Fixed-Size Journal Files

From a management perspective, once you set up a fixed-size journaling system, there is little operator intervention required. When you use multiple, fixed-size journal files, after-image journaling automatically switches to the next available journal file when the current after-image journal file becomes full.

OpenVMS
VAX  OpenVMS
Alpha 

(You can be notified of this event automatically by setting the notification state. Use the `Notify` qualifier on the `RMU Set After_Journal` command.) ♦

The backup process requires minimal operator intervention because the database root file maintains these after-image journal sequence numbers:

- A backup sequence number (the sequence number of the next after-image journal file to be created)
- A recovery sequence number (the sequence number of the next journal file to be recovered)

Using these two after-image journal sequence numbers, the database root file structures manage the journaling system and recovery of the entire set of journal files.

9.6.8.1 Automatic and Manual Backup Operations

As soon as the full journal file is off line, you can back up the journal file to tape media manually (using the RMU Backup After command) or enable the ABS process to back up journal files automatically.

Automatic journal file backup operations are advantageous because Oracle RMU performs the backup operation without requiring human intervention. The backup operation must write to a disk-based backup file. Ideally, each backed-up journal file is located on a separate disk from all other database files. If you cannot provide a disk for each journal file, minimally, you should locate journal backup files only on disks with sufficient space.

Note

Although the ABS process can perform a backup operation to a file across a network (include a node name in the backup file specification), it is probably faster to back up the file to a local disk and then copy the disk to the remote site.

Manual journal file backup operations are advantageous because, when given sufficient disk resources, you can perform after-image journal file backup operations on the complete set of journal files just prior to the database backup operation.

After a journal file is backed up, you can copy the backup file to tape using the OpenVMS BACKUP command or the Digital UNIX tar function. Then, delete the backed-up journal file from the disk to make the disk space available for more after-image journal backup operations. Also, you can perform after-image journal backup operations manually to tape devices.

For example, assume that you want to use three after-image journal files, an automatic backup server (ABS), an automatic AIJ log server (ALS), and you enabled AIJ notification. The following examples show how to use SQL statements to specify these options and display the results:

```
SQL> -- Specify the journal backup file name
SQL> -- for the first journal.
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ALTER JOURNAL JOURN_1
cont> BACKUP FILENAME DISK21$:[JOURNAL]BKUPJOURN_1; ❶
SQL> -- Reserve four journals.
```

```

SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> RESERVE 4 JOURNALS;
SQL> -- Add two more journals, allocations, and backup file names.
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD JOURNAL JOURN_2 FILENAME DISK12$:[JOURNAL]JOURN_22
cont> ALLOCATION 1024 BLOCKS
cont> BACKUP FILENAME DISK22$:[JOURNAL]BKUPJOURN_2
cont> ADD JOURNAL JOURN_3 FILENAME DISK13$:[JOURNAL]JOURN_32
cont> ALLOCATION 1024 BLOCKS
cont> BACKUP FILENAME DISK23$:[JOURNAL]BKUPJOURN_3;
SQL> -- Enable the backup server as automatic.
SQL> -- Enable notification.
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> JOURNAL IS ENABLED
cont> (BACKUP SERVER IS AUTOMATIC,3
cont> NOTIFY IS ENABLED);4
SQL> -- Show the current parameters for the journals.
SQL> SHOW JOURNALS *
Journals in database with filename mf_personnel
JOURN_1
  AIJ File Allocation:          1024
  AIJ File Extent:             512
  Journal File:                DISK11$:[JOURNAL]JOURN_1.AIJ;1
  Backup filename:             DISK21$:[JOURNAL]BKUPJOURN_1.AIJ;1
JOURN_2
  AIJ File Allocation:          1024
  Journal File:                DISK12$:[JOURNAL]JOURN_2.AIJ;1
  Backup filename:             DISK22$:[JOURNAL]BKUPJOURN_2.AIJ;1
JOURN_3
  AIJ File Allocation:          1024
  Journal File:                DISK13$:[JOURNAL]JOURN_3.AIJ;1
  Backup filename:             DISK23$:[JOURNAL]BKUPJOURN_3.AIJ;1

```

```

SQL> SHOW DATABASE RDB$DBHANDLE ⑤
.
.
.
    AIJ File Allocation:          1024
    AIJ File Extent:             512
.
.
.
    Journal is Enabled
    Journal File:      JOURN_1
    Backup Server:    Automatic
    Log Server:       Manual
    Overwrite:        Disabled
    Notification:     Enabled
.
.
.
Journals in database with filename mf_personnel
    JOURN_1
    JOURN_2
    JOURN_3

```

- ① Check that your original journal file has a backup journal file name; if not, add one using the SQL ALTER DATABASE ALTER JOURNAL statement.
- ② Add two additional journal files on separate disk drives
- ③ Specify the ABS process as automatic.
- ④ Enable after-image journal file notification.
- ⑤ Display the results of your changes.

Each time there is a journal file switchover, the operator is notified and an automatic backup operation of the full journal file occurs. For example, a display of the header information shows you the current status of journaling for each journal file after a switchover occurs.

```

AIJ Journaling...
- After-image journaling is enabled
- Database is configured for 5 journals
- Reserved journal count is 5
- Available journal count is 3

```

```

- Journal switches to next available when full
- 1 journal has been modified
- 2 journals can be created while database is active
- Journal "JOURN_2" is current❶
- All journals are accessible
- WARNING: Non-journalled database modifications have been made❷
  - Recovery (roll-forward) may not be possible
  - Full database backup is required
- Shutdown time is 60 minutes
- Backup Spooler is enabled
- Log server startup is MANUAL
- Operator notification is enabled for the following operators
  - Central
- Journal overwrite is disabled
- AIJ Cache on Electronic disk is disabled
- Default AIJ journal allocation is 512 blocks
- Default AIJ journal extension is 512 blocks
- Default AIJ journal initialization is 512 blocks
- Current roll-forward sequence number is 1
- Current backup sequence number is 1
- Database backup has not been done
  - Journaling was either enabled or new journals reserved
  - Full database backup is recommended
- Last Journal backed up was "JOURN_1"❸
  - Backup sequence number was 0❹
- Next journal to be backed up is "JOURN_2"❺
  - Backup sequence number is 1❻
- AIJ Journal "JOURN_1"❼
  - Filename is DISK11$:[JOURNAL]JOURN_1.AIJ;2
.
.
.
- AIJ Journal "JOURN_2"
.
.
.

```

- ❶ JOURN_2 is the current journal file.
- ❷ A Warning message instructs you to perform a full database backup operation because the database is not recoverable currently.
- ❸ JOURN_1 is backed up.
- ❹ The JOURN_1 backup sequence number was 0.
- ❺ The next journal file to be backed up is JOURN_2.
- ❻ JOURN_2 has backup sequence number 1.

- ⑦ The display continues, providing complete information about JOURN_1 and all other journal files.

As shown in callout ①, reserving journal files is not a journaled event so you must back up the database to ensure that it can be recovered. To immediately back up the current journal file, you can manually switch journal files using the `Switch_Journal` qualifier on the `RMU Set After_Journal` command, and then performing a quiet-point backup operation on the full journal file. The following code example shows how to perform these operations:

```
$ RMU/SET AFTER_JOURNAL /SWITCH_JOURNAL MF_PERSONNEL
$ RMU/BACKUP MF_PERSONNEL MFPERS_101293
```

A display of the header information, as shown in the following examples, indicates more current status for journaling and the journal files:

```
.
.
.
AIJ Journaling...
- After-image journaling is enabled
- Database is configured for 5 journals
- Reserved journal count is 5
- Available journal count is 3
- Journal switches to next available when full
- 1 journal has been modified
- 2 journals can be created while database is active
- Journal "JOURN_1" is current①
.
.
.
- Current backup sequence number is 2
- Database backup AIJ sequence number is 2
- Last Journal backed up was "JOURN_2"②
  - Backup sequence number was 1③
- Next journal to be backed up is "JOURN_1"
  - Backup sequence number is 2④
- AIJ Journal "JOURN_1"
  - Filename is DISK11$:[JOURNAL]JOURN_1.AIJ;2
  - Default AIJ filename is "JOURN_1"
  - Journal is current
  - Backup sequence number is 2④
.
.
.
```


The warning message about not being able to recover your database because of unrecoverable changes is gone.

- ❶ JOURN_1 is the current journal file.
- ❷ JOURN_2 was the last journal file to be backed up.
- ❸ JOURN_2 has a backup sequence number of 1.
- ❹ JOURN_1 has a backup sequence number of 2.

9.6.8.2 File Management

As you devise a backup strategy, you need to determine the following:

- How often you want to schedule full database backup operations
- How many journal files you usually accumulate between full backup operations

If you are planning database backup operations on a regular basis, for example, once a week, and estimate that you will be filling and backing up journal files twice a day, you could use a manual backup method (such as a DCL command procedure) to keep track of the journal backup files that contain information and the duration of time covered by each backup file.

To do this, use the `Edit_Filename=(Options)` qualifier with the `RMU Backup After_Journal` command to specify that a date, time, and other values be applied to the file name to make it more meaningful and to make the management of the files easier.

For example, by specifying the following options, Oracle RMU supplies the date, time, and journal sequence number of each backed up journal file. The advantage of this file-naming approach is easier recovery management because you must know in what order to apply the backed up journal files after you restore your database.

```
$ RMU/BACKUP/AFTER_JOURNAL -  
_$_ /EDIT_FILENAME=("_",MONTH,DAY_OF_MONTH,YEAR,"_",HOUR,MINUTE,"_",SEQUENCE) -  
_$_ DISK21$:[JOURNAL]JOURN_1
```

A directory of `DISK21$:` that contains the `JOURN_1` journal backup file displays a more meaningful file name. The `SEQUENCE` value tells you the backup sequence number of the journal backup file which is essential information for recovery purposes.

```
JOURN_1_10121996_1913_8.AIJ:1
```

Depending on how frequently the after-image journal file switchover occurs, you may want to perform an OpenVMS BACKUP operation or Digital UNIX tar function to tape on a regular basis for each disk containing a journal backup file.

You could write a procedure to further automate this process if you had a tape drive dedicated solely for this purpose. For example, the procedure could look for new file names on the disk during different periods of the day. Some operator intervention may be required to respond to prompts.

You could perform backup operations either manually or automatically with a user-written procedure. Backup operations cannot be automated with the after-image journal backup server (ABS) because the ABS process records only the backup file name that you specify for each journal file.

9.6.9 Writing a Customized Backup Procedure

You might want to create a procedure that determines when journal files switch and performs a semiautomatic after-image journal backup operation.

OpenVMS OpenVMS
VAX Alpha

For example, Example 9-4 is a sample DCL procedure that detects when the after-image journal file switchover occurs, and then automatically backs up the full journal file.

Example 9-4 uses the following global process symbols to help automate the after-image journal backup operation:

- RDM\$AIJ_SEQNO—Sequence number of the last journal backup file written to tape.
- RDM\$AIJ_CURRENT_SEQNO—Sequence number of the currently active journal file. A value of -1 indicates that after-image journaling is disabled.
- RDM\$AIJ_NEXT_SEQNO—Sequence number of the next journal file that needs to be backed up.
- RDM\$AIJ_LAST_SEQNO—Sequence number of the last journal file that needed to be backed up. A value of -1 indicates that no journal file has ever been backed up.
- RDM\$AIJ_BACKUP_SEQNO—Sequence number of the last journal file backed up by the backup operation. A value of -1 indicates that this process has not yet backed up a after-image journal file.
- RDM\$AIJ_COUNT—Number of available after-image journal files.

Example 9-4 Automated Custom Backup Procedure

```
#!/+++++
#!/ User-written Backup Server command script for Oracle Rdb on OpenVMS
#!/
#!/ If no parameters are supplied, questions are asked:
#!/ P1 = Database rootfile specification.
#!/ P2 = AIJ backup filename specification.
#!/ P3 = Termination date/time
#!/-----
#!/
$ on ERROR then goto exit_backup
$ on WARNING then goto exit_backup
$ on CONTROL_C then goto exit_backup
$ on CONTROL_Y then goto exit_backup
#!/
#!/ Get the database name
#!/
$ if p1 .eqs. "" then inquire p1 "_database"
$ DB_NAME = "'p1'"
#!/
#!/ Get the backup filename
#!/
$ if p2 .eqs. "" then inquire p2 "_backup"
$ BACKUP_NAME = "'p2'"
#!/
#!/ Get the termination date/time
#!/
$ if p3 .eqs. "" then inquire p3 "_until"
$ UNTIL_TAD = "'p3'"
$ UNTIL_DATE = f$cvtime(UNTIL_TAD)
#!/
#!/ Get the initial backup context information. This initializes 4 interesting
#!/ process-global symbols:
#!/ RDM$AIJ_COUNT      - Number of available AIJ journals
#!/ RDM$AIJ_CURRENT_SEQNO - Sequence# of currently active AIJ journal
#!/ RDM$AIJ_NEXT_SEQNO  - Sequence# of next AIJ journal needing backup
#!/ RDM$AIJ_LAST_SEQNO  - Sequence# of last AIJ journal needing backup
#!/
$ RMU /SHOW AFTER_JOURNAL /BACKUP_CONTEXT /OUT=NL: 'DB_NAME

#!/
#!/ Make sure multiple AIJ Journals exist.
#!/
$ if RDM$AIJ_COUNT .eq. 1
$ then
$ rmu /backup /after_journal /quiet /log -
/continuous /until="'UNTIL_TAD'" /interval=60 -
'DB_NAME 'BACKUP_NAME
$ goto exit_backup
$ endif
```

(continued on next page)

Example 9–4 (Cont.) Automated Custom Backup Procedure

```
#!/
#!/
#!/ Are there any journals ready to be backed up?
#!/
$ continue_backup:
$ sho symbol RDM$AIJ*
#!/
#!/ Check the termination date/time
#!/
$ CUR_DATE = f$time()
$ CUR_DATE = f$cvtime(CUR_DATE)
$ if CUR_DATE .gts. UNTIL_DATE then goto exit_backup
#!/
#!/ Make sure after-image journaling is enabled. Do this inside the backup loop
#!/ in case after-image journaling is disabled while this script is executing.
#!/
$ if RDM$AIJ_CURRENT_SEQNO .eq. -1
$ then
$ write sys$output "AIJ journaling for database ''DB_NAME' is disabled"
$ goto exit_backup
$ endif
#!/
#!/ Is there an AIJ journal ready for backup?
#!/
$ if RDM$AIJ_NEXT_SEQNO .le. RDM$AIJ_LAST_SEQNO
$ then
$ RMU /BACKUP /AFTER_JOURNAL /LOG -
/sequence=('RDM$AIJ_NEXT_SEQNO, 'RDM$AIJ_LAST_SEQNO) /wait -
'DB_NAME 'BACKUP_NAME
$ endif
#!/
#!/ Wait for 1 minute before updating our backup context again
#!/
$ wait 00:01:00.00
$ rmu /show after_journal /backup_context /out=nl: 'DB_NAME
$ goto continue_backup
#!/
$ exit_backup:
$ verify = f$verify(verify)
$ exit
```

◆

9.7 Journal Switchover

Journal file **switchover** occurs when journaling switches from logging database modifications in a (full) journal file to a new (empty) fixed-size journal file.

Typically, after-image journal file switchover occurs because the current journal file is full. As a journal file fills with modifications, logging automatically switches to unmodified journal files so that the I/O operations continue logging journal records. Journal file switchover also occurs for other reasons, such as when you manually switch journal files or if there is a hardware failure.

Switching journal files usually occurs transparently to database users and applications. As journal files fill up and successfully switch over to the next unmodified journal file, checkpointing occurs automatically for each database user. The purpose of performing a checkpoint operation is to save the status each time switchover occurs and to:

- Avoid a long interval of time between checkpoints
- Prevent a long transaction from spanning several journal files

Either of these situations could lead to a condition similar to running out of disk space.

During a journal file switchover, transaction processing can continue without interruption as long as there are sufficient unmodified journal files and journal slots available. New, unmodified after-image journal files become available when one of the following events occurs:

- A modified journal file is backed up, thus becoming available for journaling again
- A new after-image journal file is added

However, if extra, unmodified journal files are unavailable, Oracle Rdb suspends all journal file processing until an unmodified journal file becomes available. It is a serious situation when journaling suspends because the journal file switchover cannot occur, but it is completely avoidable if you follow the recommendations in Section 9.7.2.

9.7.1 Reasons Why After-Image Journal File Switchover Suspends

Although there are a number of mechanisms to help you ensure that after-image journaling never suspends, it is possible for suspension to occur for the following reasons:

- The complete set of after-image journal files is unavailable when an additional journal file is needed
- The after-image journal file backup strategy is inappropriate for your journaling system
- The after-image journal file allocation size is inadequate

When transactions are waiting to commit and all database modification information needs to be written to a journal file, Oracle Rdb suspends all after-image journaling activities until a new journal file is available. Database activity must not continue until a journal file is available because all database modifications are required for future after-image rollforward operations.

Warning

Once after-image journal file switchover is suspended, any process failure, including failures caused by explicitly entering a command to stop the process initiates an immediate database shutdown. Database shutdown is necessary because the database recovery process (DBR) cannot write either the pending after-image journal file data or the transaction rollback record to the after-image journal file. Once the database is frozen, no database locking operations are allowed. This is known as a deadlock on after-image journal file switchover.

Any database operation (such as the RMU Backup command or the RMU Backup After_Journal command) can cause the database to shut down if the backup operation is aborted. Avoid aborting backup operations to minimize the possibility of database shutdown.

9.7.2 Avoiding Journal File Switchover Suspension

You must implement journal file switchover carefully to ensure that there are an adequate number of journal files and reserved journal slots available.

Use the following strategies to plan for contingencies and unexpected events, and avoid suspended after-image journaling operations:

Reserve at Least One Unused After-Image Journal Slot

Reserve sufficient journal slots to allow for after-image journaling as well as after-image journal file backup operations. Always have at least one unused journal slot and more if you have enough disk space (each slot requires two blocks in the database root file). Reserving a journal slot does not create an after-image journal file.

If you run out of reserved journal slots, you cannot add more slots while the database is on line and active. Increasing the number of reserved journal slots requires that you take the database off line.

Note

If a process is stalled because it is waiting for more journal files to be added, you cannot increase the number of journal slots even if the database is off line.

Consider reserving one or two extra disks and extra journal slots for failure recovery. In case a journaling disk goes down, you can easily add another journal file in an online operation as long as there are extra slots and a disk available.

Add a Minimum of Three After-Image Journal Files

Plan ahead to correctly estimate the number of journal files required for the database activity typical for your application. Oracle Corporation recommends that you add a minimum of three journal files.

If you fill all available unmodified journal files and have no more reserved journal files, and you cannot add another journal file to allow journal file switchover, after-image journaling operations suspend after 60 minutes (the default).

Specify the Appropriate Allocation Size for the After-Image Journal Files

Proper sizing of the after-image journal files is extremely important to the overall smooth operation of the application. Care should be taken when the application transactions are large or when high-transaction throughput is important. Remember that each after-image journal file should contain 1200 blocks per number of database nodes. (See Section 9.4.3.)

Enable AIJ Operator Notification

Enable after-image journaling operator notification using the NOTIFY IS ENABLED clause on the SQL ALTER DATABASE statement.

For example, when journaling activity switches to the last unmodified after-image journal file, the system operator receives the following broadcast message:

```
%%%%%%%% OPCOM 17-MAY-1996 13:49:56.41 %%%%%%%%%
Message from user ORACLEUSER on ALPHA
Oracle Rdb Database DB$DISK:[DIRECTORY]MF_PERSONNEL.RDB;1
Event Notification
Last unmodified AIJ journal has been selected
```

Also, see the OpenVMS documentation for more information about using DCL commands (such as SET TERMINAL/BROADCAST, SET BROADCAST=OPCOM, and REPLY /ENABLE=*operator-classes*). ♦

Enable an Emergency After-Image Journal File

You can completely avoid journal file switchover suspension by using the following logical name or configuration parameter:

- For OpenVMS systems, define the RDM\$BIND_ALS_CREATE_AIJ logical name in the LNM\$SYSTEM_TABLE logical name table.
- For Digital UNIX systems, define the RDB_BIND_ALS_CREATE_AIJ configuration parameter in the rdb.conf configuration file.

If the after-image journal file switchover operation becomes suspended and you have defined this logical name or configuration parameter, Oracle Rdb creates an **emergency** after-image journal file. An emergency after-image journal file is a standard journal file that the ALS process creates to avoid the journal file switchover suspension.

Note

If a process fails when the ALS creates an after-image journal file, the database does not shut down.

You must define the logical name or configuration parameter on all systems where the AIJ log server (ALS) is active using the following settings:

- Specify the default value “0” to indicate that the ALS process should not create an after-image journal file
- Specify the value “1” to indicate that the ALS process should create an after-image journal file

The ALS process creates the emergency journal file using the same directory and allocation size as for the previous journal file. If an error occurs (such as inadequate disk space), after-image journal file operations become suspended.

You can specify the location of the emergency after-image journal file using the RDM\$BIND_AIJ_EMERGENCY_DIR logical name or the RDB_BIND_AIJ_EMERGENCY_DIR configuration parameter. This logical name or configuration parameter applies to all databases on the current node.

OpenVMS OpenVMS
VAX Alpha

The device and directory you specify must not contain nonsystem concealed logical definitions. ♦

When the ALS process creates the emergency journal file, it notifies you via the operator notification facility. In addition, you can use the RMU Dump Header command or the RMU Show Statistics command to identify emergency after-image journal files.

The following example shows the AIJ Journal Information screen. In the example, notice the name and file specification of the emergency after-image journal is EMERGENCY_X where the X represents a 16-character unique name:

```
Node: ALPHA   Oracle Rdb V7.0-00 Performance Monitor   20-MAY-1996 13:24:27
Rate: 1.00 Second   AIJ Journal Information   Elapsed: 02:12:56.32
Page: 1 of 1   KODH$:[R_ANDERSON.WORK.ALS]MF_PERSONNEL.RDB;1   Mode: Online
-----
Journaling: enabled   Shutdown: 60   Notify: enabled   State: Accessible
ALS: Manual   ABS: disabled   ACE: disabled   FC: enabled   CTJ: disabled

After-Image.Journal.Name..... SeqNum AIJsize CurrEOF Status. State.....
JOURN_1                0 *BACKUP NEEDED* Written Accessible
JOURN_2                1 *BACKUP NEEDED* Written Accessible
JOURN_3                2 *BACKUP NEEDED* Written Accessible
JOURN_4                3 *BACKUP NEEDED* Written Accessible
JOURN_5                4 *BACKUP NEEDED* Written Accessible
JOURN_6                5 *BACKUP NEEDED* Written Accessible
EMERGENCY_009A29D3D0FC262E 6    512        2 Current Accessible
Available AIJ slot 1
Available AIJ slot 2
Available AIJ slot 3
Available AIJ slot 4
-----
Bell Exit Help Menu >next_page <prev_page Refresh Set_rate Write Zoom !
```

Note

You cannot remove the emergency status of an emergency after-image journal file.

Set an Adequate Time Interval Before Database Shutdown

Set the time interval before database shutdown long enough to allow you to perform an after-image journal file backup operation and make a journal file available.

The default is 60 minutes. You can specify using the `Shutdown_Timeout` qualifier on the `RMU Set After_Journal` command as shown in the following example:

```
$ RMU/SET AFTER/SHUTDOWN=120/LOG MF_PERSONNEL
```

Once the database begins to shut down, you cannot change the shutdown value.

If you cannot back up the journal file during the shutdown time, Oracle Rdb shuts down the database. Then, the process initiating the after-image journal file shutdown produces a bugcheck dump to provide an analysis of the problem. In this case, recovery from an after-image journal file failure is almost impossible.

Automate the Backup Operation

You should enable the AIJ Backup Server (ABS) or write your own procedure to automatically back up modified journal files as soon as journal file switchover occurs:

- **Enable the AIJ Backup Server (ABS) Process**
The ABS process backs up to disk only, so it is possible for the backup operation to fail if there is insufficient disk space. You can avoid this situation by closely monitoring the disk.

Note

If your database is enabled with the Commit To Journal optimization, shutdown is immediate because the after-image journal file backup operation is impossible.

Section 9.6.8.1 provides more information about automatically backing up your journal files.

- **Write a Customized Backup Procedure**

You might want to write a procedure that determines when journal files switch and perform a semiautomatic after-image journal file backup operation.

Section 9.6.9 provides a sample DCL procedure that detects when the after-image journal file switchover occurs and then automatically backs up the full journal file.

Use the RMU Show Statistics Screen

Review the information on the AIJ Information and Stall Messages screens and, if necessary, modify your journaling procedures based on the state of the after-image journal files.

The AIJ Information screen in Example 9–5 provides real-time information about the state of the after-image journaling subsystem. For example, assume that Example 9–5 displays information for a suspended after-image journal file.

Example 9–5 After-Image Journal File Information Statistics Screen

```
Node: ALPHA      Oracle Rdb V7.0-00 Performance Monitor 21-MAY-1996 11:15:04
Rate: 1.00 Second  AIJ Information                               Elapsed: 00:05:59.77
Page: 1 of 1  DISK$:[ORACLEUSER.WORK.AIJ]MF_PERSONNEL.RDB;1  Mode: Online
-----
Journaling: Enabled  ❶Shutdown: 56  Notify: Enabled S
ALS: Manual      ABS: Disabled  ACE: Disabled  FC: Enabled  CTJ: Enabled
After-Image.Journal.Name..... SeqNum AIJsize CurreEOF Status. State.....
❷JOURN_1                               ❸0 *BACKUP NEEDED* Written Access
JOURN_2                               1 *BACKUP NEEDED* Written Accessible
❹JOURN_3                               2    513  ❺512 Current Access
Available AIJ slot 1
Available AIJ slot 2
```

Example 9–5 provides an extremely powerful tool for database management because it displays:

- ❶ When an after-image journal file switchover is suspended, the Shutdown field is highlighted and the number of minutes remaining until database shutdown is displayed. In this case, 56 minutes remain until the database is shut down.
- ❷ The current state of each after-image journal file. In this example, the screen indicates that journal file JOURN_1 has been modified (status is Written) and needs to be backed up *BACKUP NEEDED*. Journal file JOURN_2 has been modified but has not been backed up.
- ❸ Journal sequence numbers; in this case, the journal sequence number is 0.

- ④ When the last available journal file is to be written. Notice that all journal files except one have been written but the journal files have not been backed up.
- ⑤ Used space (512 blocks) in the current journal file. The difference between the values in the AIJsize and CurrEOF fields indicates how much space is available until journal file switchover occurs.

The RMU Show Statistics Stall Messages Screen in Example 9–6 identifies the cause of the after-image journal file switchover suspension. A suspended journal file switchover is considered a stall, and the stall description identifies both the problem and the number of minutes remaining until database shutdown occurs. This stall is not associated with a lock however. The stall is waiting for the state of a database structure to change.

Example 9–6 Active User Stall Messages Screen

```
Node: ALPHA Oracle Rdb V7.0-00 Performance Monitor 21-MAY-1996 11:15:16
Rate: 1.00 Second Active User Stall Messages Elapsed: 00:06:11.22
Page: 1 of 1 DISK$:[ORACLEUSER.WORK.AIJ]MF_PERSONNEL.RDB;1 Mode: Online
-----
Process.ID Since..... Stall.reason..... Lock.ID.
2A401E03:1 11:15:07.93 - waiting for 1600-block unmodified AIJ (56 minut
-----
```

Also, you can examine the Active User Stall Messages screen:

- When a transaction process is hung to estimate (worst case) the number of after-image blocks needed when after-image journal file processing resumes.

In Example 9–6, Oracle RMU estimates that approximately 1600 blocks of data might be written to the after-image journal file by all nodes. In most cases, fewer blocks are written to the after-image journal file when after-image journal file processing resumes.

- During normal processing to determine approximately how many blocks have been modified in the current after-image journal file.

For example, any process that writes to the after-image journal file stalls waiting for the I/O operations to complete. This is because the I/O operations must be performed as a synchronous operation. The Active User Stall Messages screen indicates where the last I/O stall to the journal file occurred, and displays the sequence number of the current journal file.

Use the RMU Dump Header Information

Review the RMU Dump Header information and, if necessary, make appropriate modifications based on the state of the after-image journal files.

Practice Resolving Journal File Switchover

Even though you make a concerted effort to prevent an after-image journal file switchover from suspending database operations, Oracle Corporation recommends that you practice the steps needed to resolve a suspended journal file switchover condition. You can assimilate journal file switchover by entering the following command:

```
$ RMU/SET AFTER_JOURNAL/SWITCH/LOG database_rootfile
```

9.7.3 Determining If Journal File Switchover Is Suspended

Use the following methods to help you determine when after-image journal file switchover is suspended:

- **The Database Is Shut Down**

Typically, this problem goes unnoticed because it usually occurs when the database is performing batch processing. When the database is shut down, any attempt to process transactions results in Oracle Rdb terminating the image returning the AJTERMINATE error. To restart database activities, you must perform a full database backup operation.
- **Transaction Processing Hang**

This problem typically occurs during peak or high-volume periods. When a transaction process hangs, any application attempting to perform database modifications does not succeed and waits indefinitely for Oracle Rdb to return control to the application. It is possible for transactions that hang to resume processing.
- **System Operator Notification**

If you enable after-image journal notification (using the NOTIFY IS ENABLED clause on the SQL ALTER DATABASE statement), Oracle Rdb returns shutdown messages to the system operator and logs the messages in the operator-log file.

Using operator notification is an excellent method for database management because the messages automatically notify you of events that could result in after-image journaling suspension both before and after the event occurs. For example, for every I/O operation that is active when the last active after-image journal file is greater than 90 percent full, the system operators receive the following broadcast message:

```
%%%%%%%%%% OPCOM 17-MAY-1996 13:49:56.41 %%%%%%%%%%%
Message from user ORACLEUSER on ALPHA
Oracle Rdb Database DB$DISK:[DIRECTORY]MF_PERSONNEL.RDB;1
Event Notification
Last active AIJ journal is 93% full
```

This notification gives you adequate warning to start journal file switchover.

If after-image journal file switchover is suspended, the broadcast messages shown in the following example is sent to the system operators every minute until database shut down occurs or the situation is corrected:

```
%%%%%%%%%% OPCOM 17-MAY-1996 13:49:56.41 %%%%%%%%%%%
Message from user ORACLEUSER on ALPHA
Oracle Rdb Database DB$DISK:[DIRECTORY]MF_PERSONNEL.RDB;1
Event Notification
!No more available AIJ journals (56 minutes to database shutdown)
```

Note

Even if terminals are not enabled to receive operator notification, the messages are recorded in the operator log file. Consult your operating system documentation for information about using the operator log file.

- **RMU Show Statistics Screens**

Use the RMU Show Statistics screens to analyze realtime information about the state of your after-image journal files. See Section 9.7.2 for several screen examples and descriptions.

9.7.4 Resuming After-Image Journaling Operations

The main reasons that after-image journal file switchover operation suspends is because of improper after-image journal file backup operations or because you need to add another after-image journal file.

You can use either of the following commands to resolve the situation:

- **RMU Backup After_Journal**—Backs up one or more journal files
- **RMU Set After_Journal**—Adds a new journal file

Each command has trade-offs with respect to the amount of time required to accomplish the operation.

9.7.4.1 Manually Performing a Full or By-Sequence After-Image Journal File Backup

Even when journal file switchovers are suspended, you can still perform a manual full or by-sequence after-image backup operation.

As you determine whether to perform a full or by-sequence after-image backup operation, consider the following:

- A by-sequence after-image backup is essentially a noquiet-point backup operation. Therefore, it cannot be optimized.
- When performing a full after-image backup operation, the journal file switchover processing automatically resumes when you back up the first journal file.

That is, journal file switchover operations resume before the full after-image backup completes. When combined with the quiet-point capabilities of the full after-image journal backup operation, this option is most beneficial.

Performing a Full Quiet-Point Backup Operation

Perform a full after-image backup using the `Quiet_Point` or the `Noquiet_Point` qualifiers. For example:

```
$ RMU/BACKUP/AFTER_JOURNAL/QUIET_POINT MF_PERSONNEL.RDB BACKUP.AIJ
```

When you perform a quiet-point after-image journal file backup with fixed-size after-image journal files, the backup operation completes when all transactions reach the quiet point and write after-image information to the journal file being backed up. If an after-image switchover occurs while the after-image journal file backup operation is waiting for the quiet point, the backup operation cannot complete. In this situation, you must also back up the new after-image journal file.

Therefore, Oracle Corporation recommends that you examine the `RMU Show Statistics AIJ Statistics` screen to determine the average blocks written per transaction. If this value exceeds the size of the after-image journal file or journal files to be backed up, then a quiet-point after-image journal backup cannot complete. In this case, you must perform noquiet-point after-image journal backup or increase the size of the after-image journal files.

Performing a By-Sequence NoQuiet-Point Backup Operation

You can perform a by-sequence after-image journal backup operation using the `Sequence=(n[,m])` qualifier. For example:

```
$ RMU/BACKUP/AFTER_JOURNAL/SEQUENCE=(5,6) MF_PERSONNEL.RDB BACKUP.AIJ
```

To estimate how long the backup operation might take to complete, assume that the typical after-image journal backup operation runs at approximately 1000 blocks per second, not including overhead costs (such as startup). This means that it would take approximately 9 minutes to back up a 500,000 block after-image journal file.

Note

Because system performance varies widely, Oracle Corporation recommends that you perform an independent analysis of the after-image journal backup performance on your system.

9.7.4.2 Adding a New After-Image Journal File

Some situations require that you add a new journal file rather than back up the currently existing journal files. This is likely to occur when you use only two after-image journal files. For example, if there are two modified journal files and an after-image journal file switchover suspension occurs, you cannot perform a quiet-point backup operation because the transactions cannot commit until a journal file is made available.

When the only other option is to add a new after-image journal file, you must:

- Have at least one unused after-image journal slot available.
- Be sure you size the new journal file to accommodate all the pending after-image journal file modifications.

While journaling file switchover is suspended, it is possible to accumulate up to 1200 blocks of after-image journal file information per node. For instance, if five nodes are actively accessing the database at the time that after-image journal file switchover suspends, then up to 6000 blocks of after-image journal file information can be written to the newly added journal file.

Adding a journal file with an allocation size smaller than this might result in the new journal file immediately being full and requiring another switchover. This scenario could result in another immediate suspension.

Also, adding a new after-image journal file is a journaled operation requiring approximately three blocks of after-image journal file information.

See Section 9.7 for more information.

Detecting a Deadly Embrace

In addition, you cannot back up a journal file until all transactions have been committed. This situation, called a deadly embrace, is detected by Oracle RMU when it performs a backup operation on an after-image journal file. (This situation does not occur if you perform a noquiet-point after-image backup.)

The following example shows two backup operations in which the after-image journal backup operation detects the deadly embrace and reports the condition by returning the AIJNOBACKUP status:

```
ALPHA> RMU/BACKUP/AFTER/QUIET/LOG MF_PERSONNEL BACKUP.AIJ
%RMU-I-AIJBCKBEG, beginning after-image journal backup operation
%RMU-I-AIJBCKSEQ, backing up after-image journal sequence number 0
%RMU-I-LOGBCKAIJ, backing up after-image journal JOURN_1
%RMU-I-AIJNOBACKUP, AIJ contains no transactions that qualify for backup
%RMU-I-AIJBCKEND, after-image journal backup operation completed successfully

ALPHA> RMU/BACKUP/AFTER/NOQUIET/LOG MF_PERSONNEL BACKUP.AIJ
%RMU-W-DATACMIT, unjournalized changes made; database may not be recoverable
%RMU-I-AIJBCKBEG, beginning after-image journal backup operation
%RMU-I-AIJBCKSEQ, backing up after-image journal sequence number 0
%RMU-I-LOGBCKAIJ, backing up after-image journal JOURN_1
%RMU-I-LOGCREBCK, created backup file DISK$:[USER]BACKUP.AIJ;1
%RMU-I-AIJBCKSEQ, backing up after-image journal sequence number 1
%RMU-I-LOGBCKAIJ, backing up after-image journal JOURN_2
%RMU-I-AIJNOBACKUP, AIJ contains no transactions that qualify for backup
%RMU-I-AIJBCKEND, after-image journal backup operation completed successfully
%RMU-I-LOGAIJJRN, backed up 1 after-image journal at 08:26:46.50
%RMU-I-LOGAIJBLK, backed up 254 after-image journal blocks at 08:26:46.50
```

The AIJNOBACKUP as it appears in an RMU/BACKUP/AFTER/QUIET log.

The AIJNOBACKUP as it appears in an RMU/BACKUP/AFTER/NOQUIET log.

In the example, the noquiet-point after-image journal backup operation backed up several journal files before detecting the deadly embrace condition.

Note

The exact cause of the deadly embrace is not reported by the after-image journal backup utility.

Detecting Stalled Backup Operations

It is possible a backup operation that you start with the RMU Backup After_ Journal command might stall if the database is suspended. This usually occurs because there are no journal files available for the backup operation. For example, journal files might be unavailable if the oldest modified journal file contains an active checkpoint. This condition can occur if you define a sorted index on a table with many rows or insert large multimedia segmented strings.

The following example shows a backup operation that is stalled:

```
$ RMU/BACK/AFTER/LOG/NOQUIET CHECK_RDB NOQUIET1
%RMU-W-DATACMIT, unjournalized changes made; database may not be recoverable
%RMU-I-AIJBCKBEG, beginning after-image journal backup operation
%RMU-I-AIJBCKSEQ, backing up after-image journal sequence number 51
%RMU-I-LOGBCKAIJ, backing up after-image journal CHECK_AIJ1 at 14:21:55.30
%RMU-I-LOGCREBCK, created backup file DISK$USER1:[ORACLEUSER]NOQUIET1.AIJ;1
%RMU-F-AIJJRNBSY, journal CHECK_AIJ1 is busy and cannot be backed up
%RMU-I-OPERNOTIFY, system operator notification:
Oracle Rdb Database DISK$USER1:[ORACLEUSER]CHECK_RDB.RDB;1 Event Notification
AIJ manual backup operation failed
```

You can detect when the existing journal files are full or otherwise are not available for backup by looking at the following information:

- Operator messages such as the following:

```
%%%%%%%%%% OPCOM 10-MAY-1996 11:10:15.38 %%%%%%%%%%%
Message from user ORACLEUSER on ALPHA
Oracle Rdb Database DISK$USER1:[ORACLEUSER]CHECK_RDB.RDB;1
Event Notification
New AIJ journal must be added (117 minutes to database shutdown)
```

- The state of the after-image journal files with the AIJ Information in RMU Show Statistics screens. Compare the journal sequence numbers with the checkpoint numbers available through RMU Dump User command. The following example shows the output from the RMU Dump User command:

```
$ RMU/DUMP/USER CHECK_RDB
Active user with process ID 000003D6
Stream ID is 1
Monitor ID is 1
Transaction ID is 13
Recovery journal filename is "DISK2:[RDM$RUJ]CHECK_RDB$00979806C1A39FA4.RU
Read/write transaction in progress
Last AIJ checkpoint 8:2①
Transaction sequence number is 533
```

```

Active user with process ID 000003D1
Stream ID is 1
Monitor ID is 1
Transaction ID is 41
Recovery journal filename is "DISK2:[RDM$RUJ]CHECK_RDB$00979806C20547A4.RU
Read/write transaction in progress
Last AIJ checkpoint 6:4065②
Transaction sequence number is 532

```

- ① The location of the last checkpoint record for this user is in journal sequence number 8, VBN 2.
- ② The location of the last checkpoint record for this user is in journal sequence number 6, VBN 4065.

You can use the RMU Show Statistics command (AIJ Information screen) to confirm that this journal file is the oldest (lowest sequence number).

Because the oldest (lowest) checkpoint record for any user in this database is in the journal file with sequence number 6, you cannot make journal 6 or any later journal file available for either backup or overwrite operations. When user 000003D1 checkpoints again in a later journal file, then journal 6 becomes available for backup or to be written over.

The following example shows the operator message requesting that you add a journal file rather than attempt a backup operation:

```

%%%%%%%%%% OPCOM 10-MAY-1996 11:10:15.38 %%%%%%%%%%%
Message from user ORACLEUSER on ALPHA
Oracle Rdb Database DISK$USER1:[ORACLEUSER]CHECK_RDB.RDB;1
Event Notification
New AIJ journal must be added (117 minutes to database shutdown)

```

If journaling is suspended and the oldest active checkpoint is in the oldest journal file (smallest sequence number), the RMU Backup After_Journal command hangs if you try to backup the journal files. Confirm the oldest active checkpoint using the RMU Dump User command and compare it to the current oldest journal sequence number.

- Review the RMU Show Statistics Stall Messages screen to determine whether to backup an existing after-image journal file or add a new one. The Stall Messages screen indicates the number of after-image journal file blocks that are required to complete the after-image journal file switchover operation. If the number of blocks displayed is larger than the current after-image journal files, add a new after-image journal file instead of backing up the existing after-image journal files. Once you add the new after-image journal file immediately back up the existing journal files.

9.7.5 Recovering the Database

If you cannot resume after-image journal file processing before the database shuts down or if a database process terminates abnormally when a journal file switchover operation is suspended, Oracle Rdb automatically shuts down the database, terminates all active processes, and returns the AIJTERMINATE message.

Note

When the database is shut down, you can use the RMU Show Statistics command to obtain information about the state of the database. However, when you exit from the utility, Oracle RMU returns the AIJTERMINATE message.

Now, the database is unusable and it is not recoverable because after-image journal files are unavailable. Because journal file information is lost permanently, Oracle Rdb does not allow processes to attach to the database. Processes that fail produce a bugcheck dump that does not contain an exception. In place of the exception in the stack trace portion of the bugcheck dump, there is an AIJUTL\$ABORT routine. Similarly, if the ALS process is running, it bugchecks with the same AIJUTL\$ABORT routine instead of an exception.

When the database shuts down, Oracle Rdb maintains database integrity. All modifications not journaled are written directly to the database. Oracle Rdb clears all modified journal file buffers containing committed transactions that have not checkpointed. In addition, Oracle Rdb writes undo information for the current transaction to the recovery-unit journal file so that you can undo the current transaction when the suspended after-image journaling activities resume.

Warning

When the database is suspended, do not abort any process accessing the database for any reason. The recovery (DBR) process shuts down the database because pending after-image journal file data or rollback records cannot be written to the after-image journal file.

You can reactivate the database without performing a full database restore operation (from the database backup file) by following this procedure:

1. Disable after-image journaling to allow access to the database so that the subsequent commands can be processed. For example:

```
$ RMU/SET AFTER_JOURNAL/DISABLE DB.RDB
```

2. Drop (delete) all inaccessible journal files.

It is not necessary to drop all journal files, but you should drop all journal files that are marked as inaccessible (use the RMU Dump Header command to see the journal files that are inaccessible). After you delete the last inaccessible journal file, Oracle Rdb automatically reactivates after-image journaling. However, journaling remains disabled. (You re-enable journaling in step 4.) For example:

```
$ RMU/SET AFTER/DROP=(NAME=AIJ_2) DB.RDB
```

3. Add the dropped journal files back into the journaling system as soon as possible.

```
$ SQL
SQL> ALTER DATABASE FILENAME DB
cont> ADD JOURNAL JOURN_2 FILENAME aij-2-filespec;
```

4. Re-enable after-image journaling.

Remember that adding after-image journal files does not automatically enable journaling. You must enable after-image journaling either when you add journal files or as a separate operation. For example:

```
$ RMU/SET AFTER_JOURNAL/ENABLED DB.RDB
```

5. Immediately after journaling starts, back up the database.

Note

The database is unrecoverable until you perform a full database backup.

```
$ RMU/BACKUP DB.RDB DB_FULL.RBF
```

9.8 Optimizing After-Image Journaling Performance

To improve database performance, consider using some of the following methods to optimize I/O operations when after-image journaling is enabled:

Optimization	SQL Clause	Description
Enable the AIJ log server (ALS)	LOG SERVER IS AUTOMATIC LOG SERVER IS MANUAL	For high transactions-per-second (TPS) systems, this optimization reduces locking and I/O operations to the after-image journal file
Enable the AIJ cache	CACHE FILENAME <i>cache-file-spec</i>	For an electronic disk device, this optimization reduces wait time for a transaction during the commit operation and permits you to implement a fast electronic device as a write-through cache
Enable the fast commit option	FAST COMMIT IS ENABLED	For update-intensive applications, this optimization eliminates I/O operations to recovery-unit journal files and storage area files, and postpones writing data modifications to the after-image journal file and the database root file
Enable the commit-to-journal option	COMMIT TO JOURNAL	For update-intensive applications, enable this optimization in conjunction with the fast commit option to further enhance performance by eliminating the majority of I/O operations to the database root file

You can enable these optimizations by including the appropriate clauses on either the SQL ALTER DATABASE or SQL CREATE DATABASE statement.

Note

You *must* use after-image journaling whenever you enable either the fast commit or the commit-to-journal optimizations. Furthermore, you must use the fast commit option when you enable the commit-to-journal option.

When you use these optimizations, application performance is improved compared to the same application running without after-image journaling enabled. Fast commit processing optimizes commit processing at the expense of recovery processing time. However, you can tune or reduce the recovery time by setting shorter or optimum checkpoint intervals.

The following sections briefly describe how the fast commit and commit-to-journal optimizations affect after-image journaling and journal file backup procedures. For complete information about using these optimizations, refer to the:

- *Oracle Rdb7 Guide to Database Performance and Tuning* for more information about optimizing Oracle Rdb after-image journaling performance
- *Oracle Rdb7 SQL Reference Manual* for more information about specifying the appropriate SQL statements and clauses

9.8.1 Fast Commit Processing

When transactions without the fast commit option execute a COMMIT statement, Oracle Rdb writes the data modification to the recovery-unit journal file, the storage area files, the after-image journal file, and the database root file on disk. Fast commit processing enhances journaling performance by delaying how frequently data is written to disk. Instead of writing data to the disk as soon as a transaction commits, fast-commit processing keeps the updated database pages in a buffer pool. That is, database write I/O operations are performed asynchronously.

Checkpointing

Oracle Rdb writes updated pages from the buffer pool to disk when a user-specified threshold (called a checkpoint) is reached. A checkpoint occurs when a process detaches from the database, the buffer pool overflows, or metadata changes are performed. When the checkpoint occurs, all the updated pages for multiple transactions are written to disk.

The write operation to the after-image journal file on disk is triggered when each user process writes a checkpoint to the after-image journal file. When a checkpoint occurs, all updated pages are written back to disk. Between checkpoints, the pages are written to disk only if the buffer or database page is flushed because:

- A user needs to access the after-image journal file buffer
- A user requests data on the data page
- A user enter the RMU Checkpoint command

Automatic Database Recovery

If there is a database failure, or if a user process fails, the fast commit processing provides *redo* capability; that is, the fast commit option automatically begins re-doing all committed transactions since the last checkpoint. You do not need to enter the RMU Recover command to begin database recovery operations when the fast commit option is enabled.

If a user process fails and the database recovery (DBR) server process starts up, the DBR process:

1. Reads the after-image journal file and reapplies all the committed transactions starting with the failed user's checkpoint record. (This works because after-image journal file data is always flushed, including at the end of the transaction, unlike the live data pages.)
2. Reads the recovery-unit journal file to roll back any data that was written to the after-image journal file but was not committed to the database.

9.8.1.1 Changes to the Journal File

Fast commit processing requires that you enable after-image journaling for recovery purposes. This is because Oracle Rdb writes committed transactions only to after-image journal file buffers until a checkpoint triggers a write I/O operation of the journal file buffer to disk. If a subsequent transaction fails, you must redo all the previous transactions back to the last checkpoint because the updated pages have not been written to disk.

To reprocess these transactions, Oracle Rdb uses the information written to the after-image journal file. In addition, if the recovery-unit journal file buffer has written the transaction to the database file, the failed transaction is rolled back.

The recovery-unit journal file contains the following additional information required by the DBR process when you have fast commit processing enabled:

- A transaction ID (TID) that identifies each user attached to a database. The DBR uses the TID to identify which after-image journal file record belongs to which user.
- A page sequence number (PSN) that identifies the state of a page. The after-image journal PSN is used to optimize the recovery process.

When the DBR process executes, it first recovers the updated pages not flushed to disk, and then rolls back the transaction that was current when the abnormal failure occurred.

To recover completed transactions, the DBR process executes the following steps:

1. The DBR process looks at the user's run-time user process block (RTUPB) to find where the user last checkpointed. All updates that occurred prior to the last checkpoint are already reflected in the database.
2. The DBR process records the end of the journal file. The DBR process now has a starting point (the checkpoint VBN) and an end point (the end of the journal file). This is the after-image journal file range that it must process.

3. The DBR process reads through the after-image journal file range looking for after-image journal file records marked with the TID of the failed process. If the DBR process finds a commit record for a transaction for this TID, it compares the PSN for the after-image journal file record with the PSN for the data page. If the PSNs do not match, the DBR process can ignore the redo operation. If the PSNs match, the DBR process must apply the after-image journal file record to the page.

9.8.1.2 Effects On the Journal File Backup Operations

Because the DBR process needs a set of after-image journal file records to redo a failed process, after-image journal file backup behavior is slightly different when the fast commit option is enabled. Instead of backing up the entire journal file, the journal file is backed up only to the last checkpoint record. After-image journal records after the last checkpoint record are not backed up because the recovery operation needs these records in the event the transaction must be rolled back.

In Figure 9–3, if an after-image journal backup operation occurred after P2 checkpointed but before the system failed, the backup operation records the contents in the journal file up to VBN 225 because the records after that last active checkpoint are needed for recovery.

Figure 9-3 Checkpoint Processing and the Journal File Backup Procedure

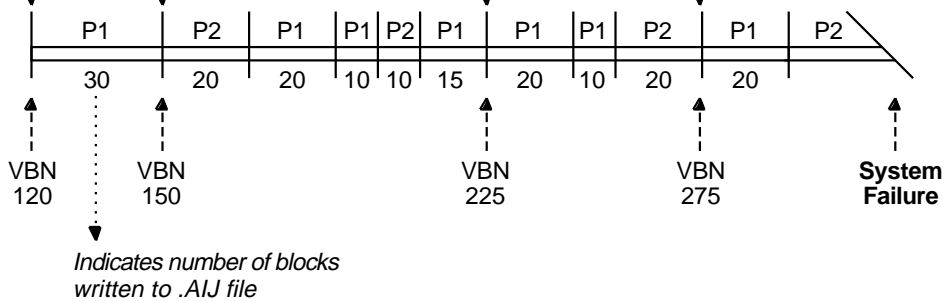
Checkpoint interval = 100 blocks

P1 attaches; writes ckpt record at VBN 120;
target VBN = 220

P2 attaches; writes ckpt record at VBN 150;
target VBN = 250

P1 ckpts (105 blocks written to AIJ); **P1** writes
new ckpt record; target VBN = 325; oldest
active ckpt now at VBN 150 (**P2**)

P2 ckpts (125 blocks written to AIJ); **P2** writes new
ckpt record; target VBN = 375; oldest active ckpt
now at VBN 225 (**P1**)



When the system fails:

- P1: transactions back to last ckpt (VBN 225) must be redone.
- P2: no transactions to redo; DBR process must undo last, uncommitted transaction.

NU-2363A-RA

After recovering committed transactions, the DBR process executes the rollback phase by rolling back the failed transaction, using the recovery-unit journal (.ruj) file. If a transaction updates a database and a user executes a ROLLBACK statement, Oracle Rdb:

1. Rolls back the effects of the current transaction
2. Flushes the data pages back to disk
3. Checkpoints

Thus, a database rollback operation triggers a checkpoint. If the transaction did not update the database, there is no need to write anything to disk and a checkpoint does not occur.

9.8.1.3 Disk Space Requirements for an Extensible Journal File

If you are using a single extensible journal file and plan to perform regular after-image journal file backup operations, plan to use twice the space of the journal file on the disk where the journal file is located. The backup process must compress and retain some fraction of the original journal file or files. This fraction can approach 100 percent of the original size. Therefore, be sure to reserve enough disk space to duplicate the maximum size of the journal file.

Oracle Rdb recommends that you schedule journal file backup operations with sufficient frequency and check the free space and journal file size periodically so that you know when you are approaching a critical situation in terms of free space. This is a good practice when using a single extensible journal file even if you do not have fast commit transaction processing enabled.

If you enter the RMU Backup After_Journal command and you find that there is insufficient disk space for the journal file, try the following options to work around this problem:

- Delete unneeded files on the disk to create sufficient space on the disk where the journal file is located.
- Temporarily disable fast commit processing and back up the journal file.
- Close the database, disable after-image journaling, enable a new after-image journal file, and perform a full database backup operation. You can open the database either before or after the full database backup operation.
- Close the database. Create a bound volume set or a striped set of disks that is large enough for the journal file and copy the journal file there. Use the RMU Set After_Journal command to drop the old journal file name and add the new journal file name (or redefine the logical name if one was used to locate the journal file), then open the database again.

9.8.1.4 Effects When Fixed-Size Journal Files Switch Over

If you are using multiple fixed-size journal files, when journaling switches to another available after-image journal file, a checkpoint occurs. The checkpoint ensures that all committed transactions are written to disk from the after-image journal file before it becomes unavailable. Then, a checkpoint record is written in the new available journal file immediately following the open record.

If an undo/redo operation becomes necessary, only the committed transactions written to the current after-image journal file since the last checkpoint are written to disk (redo) and the uncommitted transactions for the same process are undone (undo) using the user's recovery-unit journal file. This occurs because uncommitted transactions are retained in the recovery-unit journal file for each user's process. Thus, for fast commit transaction processing on a database system using multiple fixed-size journal files, when a journal file switchover occurs, Oracle Rdb automatically ensures that the undo/redo operation can always succeed with the current journal file.

9.8.2 Commit To Journal Option

For update-intensive applications, you can use the Oracle RMU Commit To Journal option to increase processing speed during commit transactions.

The Commit To Journal optimization increases commit processing speed by eliminating the majority of I/O operations to the database root file. Commit information that is normally written to the database root file is written to the after-image journal file buffer until a checkpoint occurs. Then, the buffered information is written to the after-image journal file on disk.

If you enable the Commit To Journal option, you must also:

- Enable after-image journaling
- Enable fast commit processing
- Disable snapshots (or use the Enable Deferred option on the CREATE TRANSFER (Replication Option for Rdb) statement)

9.9 Recovering Transactions from Journal Files

After a disk failure, you can use the saved backup files for the database and the after-image journal file or files to restore the database to the state it was in before the failure occurred.

Restoring a database or storage areas from a backup file produces a database that is current up to the point of the most recent database or storage area backup operation. Recovering journal files recovers transactions committed against the database up to the point of the most recent commit statement. Once the restore operation completes:

- For extensible journal files, you must use the RMU Recover command to roll forward all transactions for the database.

- For fixed-size journal files, recovery is automatic if all journal files are available. By default, once the database restore operation completes, recovery is automatic if all journal files are available.

The following sections describe how fixed-size journal files are automatically recovered and the steps involved in manually recovering a database.

9.9.1 Automatic Recovery for Fixed-Size Journal Files

For automatic recovery, any existing on-disk journal files are automatically examined to determine if the information in the journal file headers is valid. If the information is determined to be valid:

- The information from the on-disk after-image journal files is recovered into the database root data structures.
- Once the database root data structures are updated, the after-image journal file recovery operation begins and automatically rolls forward the information contained in the journal files.

For most types of database applications and especially for highly available database applications requiring 24-hour-by-7-day coverage (24x7 operations), an automatic recovery strategy is best. But this comes at the expense of requiring more disk drives for your database application.

You can disable automatic recovery using the `Norecovery` qualifier on the `RMU Restore` command. You might choose to perform a manual recovery if you need to restore some incremental backup files prior to recovering the database.

However, when you manually recover after-image journal files, you must carefully manage journal files to determine the correct order in which to apply backed-up journal files. Because the database root file structures are incapable of tracking backed-up journal files, it is essential that you use meaningful file-naming conventions to facilitate management of backed-up journal files and ensure a successful recovery operation.

9.9.2 Steps for Recovering a Database

To recover the database, you need to use the `RMU Restore` command to restore the database from the full backup file, and then use the `RMU Recover` command to roll forward the transactions stored in the after-image journal file to the database restored from a full backup file.

The following list shows you the steps for recovering the database to a known, uncorrupted state:

1. Delete the corrupt database or storage areas.

2. Restore the database, specific storage areas, or pages:
 - Restore the database using RMU Restore
 - Restore one or more storage areas using the Area and Online qualifiers on the RMU Restore command
 - Restore one or more pages using the Area, Online, and Just_Corrupt qualifiers on the RMU Restore command

If . . .	And . . .	Then . . .
Journal files are accessible on disks	Information in the journal file headers is valid	Recovery is automatic following a restore operation.
You perform a restore operation	Specify the Norecovery qualifier	Recovery becomes a manual operation. You must specify the Norecovery qualifier if you perform incremental restore operations.

3. Update the restored database, storage areas, or database pages from the last incremental backup operation by using the RMU Restore command and including the Incremental qualifier, or the Area and Online qualifiers.

If . . .	And . . .	Then . . .
You have lost one or more storage areas	Have restored just these storage areas	When you display the database root file header, these storage areas are marked as inconsistent. This information appears in the status section of the display of the database root header for each affected storage area.
You find one or more pages are corrupt in a storage area ¹	You restore just these pages	When you display the database root file header, the storage area containing these pages is marked as inconsistent, if there are journal files that must also be applied to make the area consistent.

¹As determined from a display of the corrupt page table (CPT), using the RMU Show Corrupt_Pages command.

In Example 9–7, the EMP_INFO storage area was lost. To bring the database to its most current state, you must restore the EMP_INFO storage area and then display the database root file to determine if any storage areas are marked as inconsistent. If one or more storage areas are marked as inconsistent, continue to the next step and recover the journal files in proper sequence.

Example 9-7 Restoring and Recovering a Lost Storage Area

```
! An RMU Verify operation shows that the EMP_INFO storage area cannot
! be found.
!
$ RMU/VERIFY MF_PERSONNEL
%RMU-F-OPNFILERR, error opening file DUA1:[ORION]EMP_INFO.RDA;3
%RMU-F-FILNOTFND, file not found
%RMU-E-BDAREAOPN, unable to open file DUA1:[ORION]EMP_INFO.RDA;3
for storage area EMP_INFO
%RMU-F-ABORTVER, fatal error encountered; aborting verification
!
! Or, access to the storage area returns a file-not-found error message.
!
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SELECT COLLEGE_CODE,COLLEGE_NAME,CITY,STATE,POSTAL_CODE FROM COLLEGES;
%RDB-F-SYS_REQUEST, error from system services request
-RDMS-F-FILACCERR, error opening storage area file DUA1:[ORION]EMP_INFO.RDA;3
-RMS-E-FNF, file not found
!
! The EMP_INFO storage area is lost. The EMP_INFO storage area must be
! restored. A restore operation automatically applies .aij files if they
! are available, modified, and on disk.
!
RMU/RESTORE/NOCCD_INTEGRATE/AREA/LOG MFPERS.RBF EMP_INFO
%RMU-I-RESTXT_04, Thread 1 uses devices DUA1:
%RMU-I-LOGRESSST, restored storage area DUA1:[ORION]EMP_INFO.RDA;1
%RMU-I-LOGRESSST, restored storage area DUA1:[ORION]EMP_INFO.RDA;1
%RMU-I-RESTXT_05, rebuilt 1 space management page
%RMU-I-RESTXT_06, restored 0 inventory pages
%RMU-I-RESTXT_07, rebuilt 0 logical area bitmap pages
%RMU-I-RESTXT_08, restored 30 data pages
%RMU-I-RESTXT_01, Initialized snapshot file DUA2:[ORION]EMP_INFO.SNP;1
%RMU-I-LOGINIFIL, contains 10 pages, each page is 2 blocks long
%RMU-I-AIJWASON, AIJ journaling was active when the database was backed up
%RMU-I-AIJRECFUL, Recovery of the entire database starts with AIJ file
sequence 0
%RMU-I-AIJRECARE, Recovery of area EMP_INFO starts with AIJ file sequence 0
%RMU-I-AIJBADAREA, inconsistent storage area DUA1:[ORION]EMP_INFO.RDA;1
needs AIJ sequence number 0
%RMU-I-LOGRECDB, recovering database file DUA3:[ORION]MF_PERSONNEL.RDB;1
```

(continued on next page)

Example 9-7 (Cont.) Restoring and Recovering a Lost Storage Area

```
%RMU-I-AIJAUTOREC, starting automatic after-image journal recovery
%RMU-I-LOGOPNAIJ, opened journal file DUA11:[ORION]MFPERS.AIJ;1
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-LOGRECOVR, 2 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 2 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCEC, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence
number needed will be 1
%RMU-I-LOGOPNAIJ, opened journal file DUA12:[ORION]MFPERS1.AIJ;1
%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed
%RMU-I-LOGRECOVR, 2 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCEC, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence
number needed will be 2
%RMU-I-LOGOPNAIJ, opened journal file DUA13:[ORION]MFPERS2.AIJ;1
%RMU-I-AIJONEDONE, AIJ file sequence 2 roll-forward operations completed
%RMU-I-LOGRECOVR, 2 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCEC, database recovery completed successfully
%RMU-I-AIJALDONE, after-image journal roll-forward operations completed
%RMU-I-LOGSUMMARY, total 6 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
%RMU-I-LOGSUMMARY, total 2 transactions ignored
%RMU-I-AIJSUCCEC, database recovery completed successfully
%RMU-I-AIJGOODAREA, storage area DUA1:[ORION]EMP_INFO.RDA;1 is now consistent
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence
number needed will be 2

!
!=====
! If you specify NoRecovery in the restore operation, no journal files
! are applied automatically.
!
$ RMU/RESTORE/NORECOVERY/AREA/LOG MFPERS_FULL.RBF EMP_INFO
%RMU-I-RESTXT_04, Thread 1 uses devices DUA1:
%RMU-I-LOGRESSST, restored storage area DUA1:[ORION]EMP_INFO.RDA;3
```

(continued on next page)

Example 9-7 (Cont.) Restoring and Recovering a Lost Storage Area

```
%RMU-I-LOGRESSST, restored storage area DUA1:[ORION]EMP_INFO.RDA;3
%RMU-I-RESTXT_05,   rebuilt 1 space management page
%RMU-I-RESTXT_06,   restored 0 inventory pages
%RMU-I-RESTXT_07,   rebuilt 0 logical area bitmap pages
%RMU-I-RESTXT_08,   restored 30 data pages
%RMU-I-RESTXT_01,  Initialized snapshot file DUA2:[ORION]EMP_INFO.SNP;3
%RMU-I-LOGINIFIL,   contains 10 pages, each page is 2 blocks long
%RMU-I-AIJWASON,    AIJ journalling was active when the database was backed up
%RMU-I-AIJRECFUL,   Recovery of the entire database starts with AIJ file
sequence 0
%RMU-I-AIJRECFUL,   Recovery of area EMP_INFO starts with AIJ file sequence 0
!
! Display the database root file header to determine the status of the
! EMP_INFO storage area. It is marked as inconsistent (consistent
! to TSN 75). There are transactions that need to be rolled forward
! to make the EMP_INFO storage area consistent again.
!
$ RMU/DUMP/HEADER MF_PERSONNEL
.
.
.
Storage area EMP_INFO
.
.
.
Status...
  - Area is marked inconsistent
    Consistent to TSN 0:75
    Roll-forward sequence number is 0
  - Area last backed up at 2-MAY-1996 12:53:25.84
  - Area has never been incrementally restored
.
.
.
```

(continued on next page)

Example 9–7 (Cont.) Restoring and Recovering a Lost Storage Area

```
!  
! You must apply at least the first journal file manually. The remaining  
! journal files are automatically applied if they are available,  
! modified, and on disk.  
! Display the open record of each journal file to determine which is  
! AIJ sequence number 0. It is MFPERS.AIJ. This information can also  
! be obtained by displaying the database root file (.rdb) file (RMU Dump  
! command with the Header qualifier), as shown previously.  
!  
$ RMU/DUMP/AFTER_JOURNAL MFPERS.AIJ  
.  
.  
.  
  AIJ Sequence Number is 0  
.  
.  
.  
$ RMU/DUMP/AFTER_JOURNAL MFPERS1.AIJ  
.  
.  
.  
  AIJ Sequence Number is 1  
.  
.  
.  
$ RMU/DUMP/AFTER_JOURNAL MFPERS2.AIJ  
.  
.  
.  
  AIJ Sequence Number is 2  
.  
.  
.
```

(continued on next page)

Example 9–7 (Cont.) Restoring and Recovering a Lost Storage Area

```
!  
! Roll forward the MFPERS.AIJ file.  
!  
$ RMU/RECOVER/LOG/AREA=EMP_INFO MFPERS.AIJ;1  
%RMU-I-AIJBADAREA, inconsistent storage area DUA1:[ORION]EMP_INFO.RDA;3  
needs AIJ sequence number 0  
%RMU-I-LOGRECDB, recovering database file DUA1:[ORION]MF_PERSONNEL.RDB;3  
%RMU-I-LOGOPNAIJ, opened journal file DUA1:[ORION]MFPERS.AIJ;1  
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed  
%RMU-I-LOGRECOVR, 2 transactions committed  
%RMU-I-LOGRECOVR, 0 transactions rolled back  
%RMU-I-LOGRECOVR, 0 transactions ignored  
%RMU-I-AIJNOACTIVE, there are no active transactions  
%RMU-I-AIJSUCCE, database recovery completed successfully  
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence  
number needed will be 1  
%RMU-I-AIJAUTOREC, starting automatic after-image journal recovery  
%RMU-I-LOGOPNAIJ, opened journal file DUA12:[ORION]MFPERS1.AIJ;1  
%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed  
%RMU-I-LOGRECOVR, 2 transactions committed  
%RMU-I-LOGRECOVR, 0 transactions rolled back  
%RMU-I-LOGRECOVR, 0 transactions ignored  
%RMU-I-AIJNOACTIVE, there are no active transactions  
%RMU-I-AIJSUCCE, database recovery completed successfully  
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence  
number needed will be 2  
%RMU-I-AIJAUTOREC, starting automatic after-image journal recovery  
%RMU-I-LOGOPNAIJ, opened journal file DUA13:[ORION]MFPERS2.AIJ;1  
%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed  
%RMU-I-LOGRECOVR, 2 transactions committed  
%RMU-I-LOGRECOVR, 0 transactions rolled back  
%RMU-I-LOGRECOVR, 0 transactions ignored  
%RMU-I-AIJNOACTIVE, there are no active transactions  
%RMU-I-AIJSUCCE, database recovery completed successfully  
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
```

(continued on next page)

Example 9-7 (Cont.) Restoring and Recovering a Lost Storage Area

```
%RMU-I-LOGSUMMARY, total 6 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
%RMU-I-LOGSUMMARY, total 0 transactions ignored
%RMU-I-AIJSUCCEs, database recovery completed successfully
%RMU-I-AIJGOODAREA, storage area DUAL:[ORION]EMP_INFO.RDA:1 is now consistent
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence
number needed will be 3
!
! Display the database root file header to determine the status of the
! EMP_INFO storage area. It is marked consistent. This means that
! there are no additional transactions that need to be rolled forward
! to make the EMP_INFO storage area consistent again.
!
$ RMU/DUMP/HEADER MF_PERSONNEL
.
.
.
Storage area EMP_INFO
Status...
  - Area last backed up at 2-MAY-1996 12:53:25.84
  - Area has never been incrementally restored
.
.
.
!
! Verify the area, and if successful, back up the entire database.
!
```

4. Because the restore operation specified Norecovery, use the:
 - RMU Recover command to apply the transactions in the journal file to the new, now fully restored copy of the database
 - RMU Recover command with the Area and Online qualifiers to apply the transactions in specific storage areas

The result of the rollforward operation is that all transactions committed before a disk failure are reapplied to the database or specified storage areas. There may be more than one journal file to recover, depending on the use of the RMU Backup After_Journal command.

When you use the RMU Recover command to apply transactions from the after-image journal file, your journal file specification must include the device, and, if the device is a disk, you must include the directory as shown in Example 9-8.

Example 9–8 Recovering a Database and Specifying the Device and File Specification of Journal File

```
$ RMU/RECOVER DISK1$:[DATABASE.JOURNAL]MF_PERSONNEL.AIJ
```

To roll forward journal files for specific storage areas, use the RMU Recover command with the Area and Online qualifiers to perform this operation on line and still allow access to the other storage areas of the database. Section 9.9.3 describes how to determine the after-image journal sequence number for a journal file and the proper order in which to recover multiple journal files. See Example 9–7 for an example scenario of recovery by area.

If you are rolling forward journal files for one or more storage areas, you can do this operation on line and still allow access to the other pages in the storage area of the database by using the RMU Recover command with the Area, Online, and Just_Corrupt qualifiers.

Refer to the *Oracle RMU Reference Manual* for exact syntax and examples of this operation.

5. Verify the integrity of the database or specific storage areas using:

- The RMU Verify command
- The RMU Verify command with the Area qualifier

If the integrity of the database is fine, go to step 5. If the integrity check fails, see Chapter 5 to determine what to do next.

The verify operation cannot detect transaction inconsistencies in the database unless you are restoring the database by storage area. This is because at least one index is defined for a table in the restored storage area that has rows that need to be rolled forward. The index resides in a different storage area from the table.

Under these circumstances, you can use the RMU Verify command with either the All qualifier or the Index qualifier and the name of the index defined for the table in the storage area being restored and recovered.

In Example 9–9, the index verification portion of RMU Verify command with the All qualifier detects three missing rows in the COLLEGES table in the EMP_INFO storage area and returns errors fetching dbkeys and errors fetching data records. The EMP_INFO storage area was lost and then restored but no journal files were rolled forward. In this particular case, the sorted COLL_COLLEGE_CODE index is defined for the COLLEGES table that resides in a different storage area, the RDB\$SYSTEM storage area. The index has pointers to its rows, and a data fetch of these rows as part of the index verification indicates that the rows are missing. Once

the journal files are rolled forward and these three rows are recovered, the database is consistent again and index errors are resolved.

Example 9–9 Detecting Index Errors in an Inconsistent Database

```
$ RMU/VERIFY/ALL /LOG MF_PERSONNEL
.
.
.
%RMU-I-BGNNDXVER, beginning verification of index COLL_COLLEGE_CODE
%RMU-I-OPENAREA, opened storage area EMP_INFO for protected retrieval
%RMU-E-LNGTRLNDX, line 15 beyond line index on page
%RMU-E-BADDBKFET, error fetching dbkey 67:10:15
%RMU-E-ERRDATFET, error fetching data record from B-tree index node
%RMU-I-BTRNODDBK, dbkey of B-tree node for data record is 42:665:0
%RMU-E-LNGTRLNDX, line 16 beyond line index on page
%RMU-E-BADDBKFET, error fetching dbkey 67:10:16
%RMU-E-ERRDATFET, error fetching data record from B-tree index node
%RMU-I-BTRNODDBK, dbkey of B-tree node for data record is 42:665:0
%RMU-E-LNGTRLNDX, line 17 beyond line index on page
%RMU-E-BADDBKFET, error fetching dbkey 67:10:17
%RMU-E-ERRDATFET, error fetching data record from B-tree index node
%RMU-I-BTRNODDBK, dbkey of B-tree node for data record is 42:665:0
%RMU-I-NDXERRORS,      3 index errors encountered
.
.
.
```

6. Use the Online qualifier on the RMU Backup command to perform a full, online, database backup operation to a new disk or tape that specifies a new backup file name different from your previous backup copy.

Backing up your now fully restored and recovered database provides a new starting point should you need to subsequently restore and recover your database. You should change your after-image journal file just prior to issuing the backup command, as described in Section 9.6.4.

9.9.3 Manually Recovering Journal Files

When you manually recover a database, it is important to apply the journal files in the order of their relative age, oldest one first, the next oldest, and so on to the latest (or most recent) journal file, and not to leave any gaps in the recovery operation. The same is true when you manually recover backed-up journal files.

The database contains the TSN of the last committed transaction. If there are missing TSNs between those in the database and the first TSN of the journal file, then Oracle Rdb does not apply the journal file and an error message is returned. For example:

- During a restore operation, Oracle RMU automatically applies journal files in the correct sequence if they are available, modified, and located on disk (shown in Example 9–7). Also if you apply the first journal file manually, recovery of the remaining journal files is automatic as long as the remaining journal files are also available, modified, and on disk.
- During a recovery operation, if you attempt to manually apply journal files out of sequence, the recovery operation fails and no journal files are applied.

Database Reference Points

Consider the situation in which a row was added in a transaction in one after-image journal file, and then the row is later deleted and recorded in the next journal file. If the reverse happens, a row is deleted in a transaction in one journal file and then later the row is added and recorded in the next journal file. How does the database account for these transactions to ensure a successful recovery operation of the database even if the storage area has extended? And how does the database know what the last completed TSN is?

Two reference points are kept in the database:

- The after-image journal sequence number.
Each journal file or backed-up journal file contains a unique after-image journal sequence number written in the open record. The database root file maintains an after-image journal sequence number value. The after-image journal sequence number in the root file identifies the journal file needed to start rollforward operations and to ensure the correct order for applying or rolling forward each journal file.
- The committed TSN, which indicates the highest committed TSN for the database.

A value is maintained in the file and in the open record for each journal file. The highest committed TSN number shows where in the journal file or backed-up journal file to initiate the rollforward operations. Oracle Rdb preserves the correct order of transactions by permitting journal files to be recovered only in their proper order by after-image journal sequence number and by comparing the highest committed TSN number in the database root file with the highest committed TSN in the open record of each journal file.

What Happens When the Database Is Restored?

When a database is recovered, the database root file uses these two reference points to apply journal files and transactions in correct order. When the journal file is applied to the recovered database, the after-image journal sequence number and the highest committed TSN value for the database root file and the journal file's open record are compared. The journal file is recovered only when the values are the same. If the database cannot be recovered, Oracle Rdb either gives an error message right away or searches the entire journal file before returning a warning message saying that no rows can be recovered. In any case, records are not applied if the journal file is not synchronized with the database root file.

What Happens When Records Are Recovered?

When records are applied to the restored database, the time reference (that is, the committed TSN) in the journal file is updated as transactions are recovered. Thus, after recovering one journal file, the database is updated and takes on a new transaction state based on the contents of that journal file. If this is the end of the recovery operation, the database is consistent, and you can attach to the database. If not, recover the second journal file and so forth, until all journal files are recovered and the database is consistent before you attach to the database.

AIJ (Rollforward) Sequence Numbers

When you implement a fixed-size journaling system and journaling switches over to a new journal file, the after-image journal file "AIJ sequence number" is automatically incremented. That is, the first journal file is assigned AIJ sequence number 0, the second journal file is assigned AIJ sequence number 1, and so forth.

You can display the journal sequence number using the `RMU Dump After_Journal` command with the `Header` qualifier. Example 9-10 shows the sequence number as it appears in the header information for the `journ_3.ajj` journal file.

Example 9–10 Performing a Journal File Switch Increments the After-Image Journal Sequence Number

```

$ RMU/SET AFTER_JOURNAL /SWITCH_JOURNAL MF_PERSONNEL
$ RMU/DUMP/AFTER_JOURNAL JOURN_3
.
.
.
  AIJ Sequence Number is 2
.
.
.

```

Note the following when you display AIJ sequence numbers:

If . . .	Then . . .
The after-image journal sequence number is 0	If this is a fixed-size journaling system, this journal file is the first and current journal file.
The AIJ sequence number is 0	If this is an extensible-file journaling system, the journal file is always assigned an AIJ sequence number of 0.
The AIJ sequence number is greater than 0	This is a fixed-size journaling system. When you see a journal sequence number that is greater than 0, you know that this is a fixed-size journaling system, because Oracle RMU numbers the journal files starting with the number 0. In Example 9–10 this is the third of three journal files associated with this database. When a journal file switches to the third after-image journal file (in this case, JOURN_3.AIJ), its AIJ sequence number is incremented to 2.
The AIJ sequence number is –1	The journal file is part of a fixed-size group of journals and it is available and unmodified (this is not the current journal file). When a journal file is backed up and reinitialized, Oracle RMU updates the AIJ sequence number to the value –1.

To determine the AIJ sequence number for a backed-up journal file and find out the journal file name, use the RMU Dump command with the Header qualifier to display the journal file header or open record. Example 9–11 shows some of the output from a sample RMU Dump command.

Example 9–11 Displaying the AIJ Sequence Number

```
$ RMU/DUMP/HEADER MF_PERSONNEL
.
.
.
$ RMU/DUMP/AFTER_JOURNAL DISK21$:[JOURNAL]BKUPJOURN_1.AIJ
.
.
.
* Dump of After Image Journal
*   Filename: DISK21$:[JOURNAL]JOURN_1.AIJ;1
.
.
.
1/1          TYPE=0, LENGTH=510, TAD=13-MAY-1996 13:51:04.96
Database DISK$DB:[DB]MF_PERSONNEL.rdb;1
Database timestamp is 13-MAY-1996 13:29:57.99
Facility is "RDMSAIJ ", Version is 601.0
AIJ Sequence Number is 0 <-----
Last Commit TSN is 80    <-----
Synchronization TSN is 0
Type is Normal (unoptimized)
Open mode is Initial
Journal was backed up on 13-MAY-1996 14:17:17.94
Backup type is Streamed
I/O format is Block
.
.
.
```

In Example 9–11, the AIJ sequence number is 0. Therefore, the journal file named JOURN_1.AIJ is the first after-image journal file created since the last full and complete backup operation.

When you restore a database, informational messages display indicating whether after-image journaling is enabled. If so, the restore operation indicates the AIJ sequence number of the after-image journal file you should use to begin recovery.

See Example 9–7 and refer to the *Oracle RMU Reference Manual* for more RMU Recover command information and examples.

9.9.4 Optimizing Recovery Performance

The `RMU Optimize After_Journal` command provides better rollforward performance and thus greater database availability by eliminating unneeded rollback, duplicate journal file records, and sorting of journal file records. The command writes the optimized journal file to disk or tape and uses the default file extension `.oaij`.

The following list describes how Oracle RMU optimizes after-image journal files:

- Elimination of journal file records from transactions that roll back
Transactions in a journal file that roll back are not applied during the rollforward process (they are ignored). So, the journal file records from these transactions are eliminated from the optimized journal file.
- Elimination of duplicate journal file records (duplicate journal file records are journal file records that update the same database record)
During the rollforward of a journal file, duplicate journal file records cause a database record to be updated multiple times. Each successive update supersedes the previous update so that only the last or most recent update is relevant. Therefore, all but the last update to a database record is eliminated from an optimized journal file.
- Journal file records are ordered by physical database key (dbkey)
Ordering journal file records by physical dbkey improves I/O performance at rollforward time for the optimized journal file.

OpenVMS OpenVMS
VAX Alpha

Because after-image journal file optimization uses the OpenVMS Sort/Merge utility (`SORT/MERGE`) to sort journal file records, you can improve the efficiency of the sort operation by changing the number and location of the work files used by `SORT/MERGE`.

The number of work files is controlled by the `RDMS$BIND_SORT_WORKFILES` logical name. The allowable values are 2 through 10 inclusive, with a default value of 2. You can specify the location of these work files with device specifications, using the `SORTWORK` logical name. See the OpenVMS documentation set and OpenVMS online Help for more information about using `SORT/MERGE`. ♦

If improved recovery performance and database availability are important for your database application, you should also consider the following restrictions before deciding if to use optimized journal files when recovering your database:

- You cannot optimize the current journal file.

- You cannot optimize an optimized journal file.

Note

Because an optimized journal file is not functionally equivalent to the original journal file, the source journal file should *not* be discarded after it has been optimized.

- The following restrictions apply to optimized journal files with recovery operations:
 - Do not use the optimized journal files as part of by-area recovery operations (recovery operations that use the Area qualifier on the RMU Recover command).
 - Do not use the optimized journal files as part of by-page recovery operations (recovery operations that use the Just_Corrupt qualifier on the RMU Recover command).
 - Do not use the optimized journal files with the Until qualifier on the RMU Recover command. The optimized journal file does not retain enough of the information from the original journal file for such an operation.
 - Do not use the optimized journal files with a recovery operation if the database or any storage areas (or both) are inconsistent with the optimized journal file.

To work around to these restrictions, use the original, unoptimized journal file in the recovery operation instead.

- Do not optimize a journal file that contains incomplete transactions. Incomplete transactions can occur in an after-image journal file:
 - If the journal file is backed up with a noquiet-point backup operation because transactions may span journal files
 - If the previous journal file was backed up with a noquiet-point backup operation
 - If the journal file has unresolved distributed transactions

There are no workarounds to these restrictions.

The following example shows how to optimize an after-image journal file on disk:

```
$ RMU/OPTIMIZE/AFTER_JOURNAL MFPERS.AIJ MF_PERS.OAIJ
```

You can optimize a journal file by reading it from disk or tape (if it has been backed up to a tape as an OpenVMS volume) using the RMU Backup After_Journal command with the Format=Old_File qualifier.

In either case, the journal file or existing backed-up journal files on tape are in an RMS file format. By default, backed-up journal files are written to tape devices in the RMS file format when a tape device is specified. You can optimize your journal and backed-up journal files between each full and complete backup operation for improved recovery performance if desired.

Do not delete the original set of unoptimized journal and backed-up journal files if you must work around a particular restriction listed previously. ♦

See the *Oracle RMU Reference Manual* for more information about tape support and about using the RMU Optimize command.

9.10 What Causes an After-Image Journal File to Be Inaccessible

In most cases, the after-image journal file is inaccessible because the journaling operation cannot write to the journal file. The following list describes the most common reasons for inaccessible after-image journal files:

- Lack of adequate disk space on the device where the after-image journal files resides.
When the after-image journal file cannot be extended, the database shuts down. Currently, there is no mechanism available that notifies you when disk space is exhausted.
- Someone either manually deletes the after-image journal file or powers down the disk drive where the after-image journal file is located.
- Someone tries to create an after-image journal file in an attempt to get the database up and running quickly.

All of these cases occur at database run time. If any one of these situations occurs when the transaction is started, the database remains active even though the transaction cannot continue.

The following sections describe how to recover when the journal file is lost to a media or disk drive problem.

9.10.1 Recovering a Lost Extensible Journal File

Losing the journal file due to a media failure or disk drive problem is a serious problem when you are using a single extensible journal file. The next Oracle Rdb update by the application makes the database inaccessible because the journal file cannot be updated with updates applied to the database. However, Oracle Rdb handles this type of failure.

When a transaction completes normally, Oracle Rdb journals the appropriate record (either commit or rollback) to the journal file. If a transaction is terminated abnormally, the database recovery (DBR) process logs the appropriate after-image journal record (either commit or abort) to the journal file. If, for any reason, the DBR cannot log the commit or rollback record to the journal file, the database is considered corrupt and consequently becomes inaccessible.

When a journal file is lost, the active transactions are unable to write to the journal file and terminate abnormally (returning the AIJTERMINATE message). Upon abnormal transaction termination, the DBR process is invoked. However, because the DBR process cannot write to the journal file either, the database is closed. Any further attempts to attach to the database fail.

To solve the problem, you must break the chain of DBR process failures by following these steps:

1. Disable the journal file using the Disable qualifier with the RMU Set After_Journal command.
2. From the database root file, delete the reference to the lost journal file using the Drop=name=*aijname* qualifier with the RMU Set After_Journal command.
3. Create the new journal file specification by specifying the new journal name and file specification using the Add=(name=*aijname*, file=*filespec*) qualifier with the RMU Set After_Journal command.

Note

Once the DBR process successfully recovers the database, Oracle Corporation recommends that you back up the database and create a new journal file. This is necessary because all the updates reflected in the lost journal file are lost. As a result, you cannot use an earlier backup file plus some combination of journal files to recover your database. Instead, you must use the new backup file and the new journal file to restore and recover your database.

9.10.2 Recovering a Lost Fixed-Size Journal File

If you are using multiple fixed-size journal files, Oracle Rdb handles the loss of the current journal file through the automatic failover mechanism. When the current journal file is no longer available, an automatic switchover occurs to the next available journal file with no interruption to your database application.

You can be notified when the journaling state changes by setting notification to send messages to one or more operators. Use the following methods to enable after-image journal notification:

- On OpenVMS and Digital UNIX systems, use the NOTIFY IS ENABLED clause on the SQL ALTER DATABASE statement.
- On OpenVMS systems, use the Notify qualifier with the RMU Set After_Journal command.

Using the notification facility allows you to investigate the reason for a journal file switchover when it happens and take any necessary steps to correct problems.

You can add additional journal files as an online operation only if there are extra journal slots already available (reserved). If the journal file is lost, you should back up your database for the reasons previously mentioned.

9.11 Displaying the Contents of a Journal File

The RMU Dump After_Journal command displays an after-image journal file or an optimized after-image journal file in ASCII format. The RMU Recover command uses this information to update the database if a problem occurs and you must restore and recover your database.

Example 9–12 shows how to use the RMU Dump After_Journal command to examine the contents of:

- After-image journal (.aij) file
- Optimized after-image journal (.oaij) file
- After-image journal cache (.ace) file

Example 9–13 includes all entries in the after-image journal file relative to adding a new after-image journal file. The TID is 33 for this after-image journal record; the TID is used by the database recovery process to identify which after-image journal record belongs to which user.

Only one TSN, 720 is associated with this TID and the add journal file operation. Each entry is numbered in the left column with an ordered pair of numbers (for example, 1/1) representing the virtual block number (VBN) and after-image journal file buffer (AIJBUF) numbers respectively. (In Example 9–12, a seemingly empty journal file always contains an open record when you display the contents.)

Example 9–12 Displaying the Contents of an Empty Journal File

```
$ RMU/DUMP/AFTER_JOURNAL JOURN_2.AIJ
*-----
* Oracle Rdb V7.0-0                               24-MAY-1996 16:12:20.26
*
* Dump of After Image Journal
*   Filename: USER5:[TEST]JOURN_2.AIJ;1
*-----
1/1          TYPE=0, LENGTH=510, TAD=24-MAY-1996 15:56:44.42
Database USER3:[TEST]MF_PERSONNEL.RDB;1
Database timestamp is 20-FEB-1996 15:56:09.99
Facility is "RDMSAIJ ", Version is 511.0
AIJ Sequence Number is -1
Last Commit TSN is 0
Synchronization TSN is 0
Type is Normal (unoptimized)
Open mode is Initial
Backup type is Latent
I/O format is Record
$
```

In Example 9–13, the journal file has recorded all the updates to the database. (For example, updates that are recorded when you add a second journal file.)

Example 9-13 Displaying the Contents of a Journal File

```
$ RMU/DUMP/AFTER_JOURNAL MF_PERS1.AIJ
*-----*
* Oracle Rdb V7.0-0                                24-MAY-1996 16:11:29.36
*
* Dump of After Image Journal
*   Filename: USER4:[TEST]JOURN_1.AIJ;1
*-----*
1/1①          TYPE=O, LENGTH=510, TAD=24-MAY-1996 15:24:50.34
  Database USER3:[TEST]MF_PERSONNEL.RDB;1
  Database timestamp is 20-FEB-1996 15:56:09.99
  Facility is "RDMSAIJ ", Version is 511.0
  AIJ Sequence Number is 0②
  Last Commit TSN is 696
  Synchronization TSN is 0
  Type is Normal (unoptimized)
  Open mode is Initial③
  Backup type is Active
  I/O format is Record
2/2          TYPE=N, LENGTH=82, TAD=24-MAY-1996 15:56:44.83
  Database Attach Information
  PID=20C130EF:8, TID=33
  Buffer count is 20
3/3          TYPE=D, LENGTH=426, TAD=24-MAY-1996 15:56:44.83
  TID=33, TSN=720, AIJBL_START_FLG=1④
  Partial AIJBL remains
3/4          TYPE=D, LENGTH=510, TAD=24-MAY-1996 15:56:44.83
  TID=33, TSN=720, AIJBL_START_FLG=0
  Appending to partial AIJBL⑤
  Partial AIJBL remains
4/5⑥          TYPE=D, LENGTH=154, TAD=24-MAY-1996 15:56:44.83
  TID=33, TSN=720, AIJBL_START_FLG=0
  Appending to partial AIJBL
  CRAIJ: AIJID=1, LENGTH=1024
000000000000000008000000000000000000 0000 '.....'
554F4A07FFFFFFFFF0000000000000000A 0010 '.....JOU'
00000000000000000000000000000325F4E52 0020 'RN_2.....'
FFFFFFFF0000000000000000000000000000 0030 '.....'
000000000000000000000000000000000000 0040 '.....'
::: (11 duplicate lines)
575B3A34524553555F534D5642445225 0100 '%RDBVMS_USER4:[W'
4E52554F4A5D4130364244522E445241 0110 'ARD.RDB60A]JOURN'
0000000000000000000000004A49412E325F 0120 ' _2.AIJ.....'
000000000000000000000000000000000000 0130 '.....'
```

(continued on next page)

Example 9–13 (Cont.) Displaying the Contents of a Journal File

```
          :::: (12 duplicate lines)
575B3A34524553555F534D5642445227 0200  'RDBVMS_USER4:[W'
4E52554F4A5D4130364244522E445241 0210  'ARD.RDB60A]JOURN'
000000000000000000313B4A49412E325F 0220  '_.AIJ;1.....'
00000000000000000000000000000000 0230  '.....'
          :::: (28 duplicate lines)
4/6 ⑦          TYPE=C, LENGTH=18, TAD=24-MAY-1996 15:56:44.83
          TID=33, TSN=720, AIJBL_START_FLG=255
```

\$

- ① Entry 1/1 is the after-image journal file open or header record, TYPE=O.
- ② The AIJ sequence number is 0, indicating that this is the first in the series of journal files for this database.
- ③ The Open Mode can indicate these conditions:
 - Initial—Oracle RMU creates an after-image journal file either when the after-image journal file is enabled or the previous RMU Backup After_Journal command included the Quiet_Point qualifier
 - Continuation—Oracle RMU spans an after-image journal file onto another journal file because the previous after-image journal file was backed up using the Noquiet_Point qualifier
- ④ After-images of updated storage segments are journaled in Data records. A series of segment updates is called an AIJ Block “AIJBL”.
- ⑤ The AIJ disk record AIJBUF is the 510-byte (or less) record written to the disk file. The AIJ buffer record AIJBL is the actual record containing update information to the database. It can require several AIJBUF records to contain a single AIJBL buffer if the database update consists of a large number of bytes.

The RMU Dump After_Journal display might display any of these messages:

- Continuation partial AIJBL ignored
Ignore nonstarting AIJBLs. Oracle RMU write this message when the start of the AIJBL occurs in a previous after-image journal file (and the file continues in this journal file).

- **Appending to partial AIJBL**
This message indicates that the journaling I/O operation is appending to a remaining partial AIJBL buffer; this AIJBUF buffer contains only part of an AIJBL. The start of the AIJBL buffer is written in a prior AIJBUF. The complete AIJBL contains all the data-update information and may span two or more AIJBUF buffers.
- **Resizing maximum AIJBL size to !UL bytes**
Readjusts the AIJBL buffer size and allocates virtual memory if necessary. (You can run out of PGFLQUOTA or VM.) The logic is as follows:
 - Appends the new AIJBL to any remaining old AIJBL buffer
 - Loops through the AIJBL buffer, writing out any complete records
 - Shifts the remaining data down, and adjusts the remaining length
 - Returns a message if a partial AIJBL buffer remains

For example, the message “Partial AIJBL remains” indicates that the remainder of the AIJBL buffer did not fit in this AIJBUF and is continued in the following AIJBUF buffer.

⑥ Entry 4/5 shows the information displayed when a new journal file is added to the journaling system.

⑦ Entry 4/6 is a commit record for TID 33, TSN 720.

Finally, the output from an `RMU Dump After_Journal` command might show the following message:

```
Transactions from AIJ sequence number nn may span into this journal file
```

This message indicates the last after-image journal file that contains a quiet point. During recovery, the recovery operation requires the after-image journal files starting with the (*nn*) sequence number. The recovery operation requires these after-image journal files because any single after-image journal file with an AIJ sequence number subsequent to (or greater than) the (*nn*) sequence number might contain only a partial transaction. No quiet-point backup of a single after-image journal file can have transactions spanning a single after-image journal file.

9.12 Example of Database Backup, Recover, and Restore Journaling Operations

This section uses Example 9–14 to show how to modify a database from a single extensible journal file to a set of three after-image journal files and coordinate the use of the journal file and journal backup files with a full database backup operation.

See Section 8.7 for examples of restoring and recovering database pages that show how values change for entries in the corrupt page table (CPT) as pages are restored and recovered.

Example 9–14 Using Journal Files and Backup Files for a Full Database Backup Operation

```
$ ! Create the directory structure for the database files.
$
$ CREATE/DIRECTORY DISK11$:[DB.JOURNAL]
$ CREATE/DIRECTORY DISK12$:[DB.JOURNAL]
$ CREATE/DIRECTORY DISK13$:[DB.JOURNAL]
$ CREATE/DIRECTORY DISK21$:[DB.BKUP_JOURNAL]
$ CREATE/DIRECTORY DISK22$:[DB.BKUP_JOURNAL]
$ CREATE/DIRECTORY DISK23$:[DB.BKUP_JOURNAL]
$
$ ! Define system logical names for the name of each .aij
$ ! and backup .aij file. Make sure the journal file, backup
$ ! journal file, and database files are on different disks.
$ !
$ DEFINE/SYSTEM JOURN_1 -
_ $ "DISK11$:[DB.JOURNAL]JOURN_1.AIJ"
$ DEFINE/SYSTEM JOURN_2 -
_ $ "DISK12$:[DB.JOURNAL]JOURN_2.AIJ"
$ DEFINE/SYSTEM JOURN_3 -
_ $ "DISK13$:[DB.JOURNAL]JOURN_3.AIJ"
$ DEFINE/SYSTEM BKUP_JOURN_1 -
_ $ "DISK21$:[DB.BKUP_JOURNALS]BKUP_JOURN_1.AIJ"
$ DEFINE/SYSTEM BKUP_JOURN_2 -
_ $ "DISK22$:[DB.BKUP_JOURNALS]BKUP_JOURN_2.AIJ"
$ DEFINE/SYSTEM BKUP_JOURN_3 -
_ $ "DISK23$:[DB.BKUP_JOURNALS]BKUP_JOURN_3.AIJ"
```

(continued on next page)

Example 9–14 (Cont.) Using Journal Files and Backup Files for a Full Database Backup Operation

```
$ !
$ ! Create the directory structure for the database backup (.rbf) files.
$
$ CREATE/DIRECTORY DISK30$:[DB.BACKUPS]
$
$ ! Define a system logical name for the name of the
$ ! database backup disk device and directory.
$ !
$ DEFINE/SYSTEM RDB_BACKUPS DISK30$:[DB.BACKUPS]
$
$ ! Create the directory structure for the database root (.rdb) file.
$
$ CREATE/DIRECTORY DISK1$:[DB.ROOTS]
$
$ ! Define a SYSTEM logical name for the name of the
$ ! backup disk device and directory.
$ !
$ DEFINE/SYSTEM DB_DISK DISK1$:[DB.ROOTS]

$ ! You can back up the database after you enable journaling. Use the
$ ! RMU Set After_Journal command to enable after-image journaling.
$ ! The journal file is given the logical name PERS$JOUR.
$ ! Set up a three-file, fixed-size journaling system.
$ !
$ RMU/SET AFTER_JOURNAL/ENABLE /RESERVE=4 /NOTIFY=(OPER1) -
_ $ /ADD=(NAME=JOURN_1, FILE=JOURN_1, BACKUP_FILE=BKUP_JOURN_1, ALLOC=512) -
_ $ /ADD=(NAME=JOURN_2, FILE=JOURN_2, BACKUP_FILE=BKUP_JOURN_2, ALLOC=512) -
_ $ /ADD=(NAME=JOURN_3, FILE=JOURN_3, BACKUP_FILE=BKUP_JOURN_3, ALLOC=512) -
_ $ MF_PERSONNEL
$ !
$ ! Opening the database manually prevents a user from
$ ! making changes while a backup operation is running.

$ ! Open the database with restricted access to prevent users with
$ ! insufficient privileges from accessing the database.
$ !
$ RMU/OPEN/ACCESS=RESTRICTED DB_DISK:MF_PERSONNEL
$ !
$ ! Manually switch the journal files.
$ ! Manually back up the JOURN_1 .AIJ file to disk.
$ !
$ RMU/SET AFTER_JOURNAL /SWITCH_JOURNAL DB_DISK:MF_PERSONNEL
$ RMU/BACKUP/AFTER_JOURNAL/LOG DB_DISK:MF_PERSONNEL BKUP_JOURN_1
```

(continued on next page)

Example 9-14 (Cont.) Using Journal Files and Backup Files for a Full Database Backup Operation

```
$ !
$ ! Make a full backup copy of the mf_personnel database.
$ ! This command ensures that you have a copy of the database
$ ! at a known time, in an uncorrupt state.
$ !
$ RMU/BACKUP/ONLINE/LOG DB_DISK:MF_PERSONNEL.RDB -
_ $ RDB_BACKUPS:MF_PERSONNEL_FULL.RBF
$ !
$ ! The Log qualifier displays the results of the backup operation.
$ ! Now you can use SQL statements with after-image journaling
$ ! enabled.
$ !
$ ! Close the database, so you can open it again with unrestricted
$ ! access.
$ !
$ RMU/CLOSE DB_DISK:MF_PERSONNEL
$
$ ! The RMU Open command is required to open the database as
$ ! unrestricted and because the database was previously defined
$ ! with the SQL OPEN IS MANUAL statement.
$ ! Opening the database manually improves performance.
$ !
$ RMU/OPEN/ACCESS=UNRESTRICTED DB_DISK:MF_PERSONNEL

$ SQL
SQL> --
SQL> -- Invoke the database and perform some data
SQL> -- definition and storage.
SQL> --
SQL> ATTACH 'FILENAME DB_DISK:MF_PERSONNEL';
SQL> SET TRANSACTION READ WRITE;
SQL> CREATE DOMAIN NEWFIELD TEXT (10);
SQL> CREATE TABLE TABLE1
cont> (NEWFIELD NEWFIELD_DOM);
SQL> COMMIT;
```

(continued on next page)

Example 9–14 (Cont.) Using Journal Files and Backup Files for a Full Database Backup Operation

```
SQL> SET TRANSACTION READ WRITE;
SQL> INSERT T IN TABLE1 USING
cont> T.NEWFIELD = "data";
SQL> COMMIT;
SQL>
.
.
.
```

Assume a system failure occurred at this point in Example 9–14 and corrupted your original database. Delete the old database files and rebuild the database, using the procedure shown in Example 9–15.

Example 9–15 Restoring, Verifying, and Recovering a Database

```
$ ! You know that the backed up copy of the database is
$ ! not corrupt. You can use the RMU Verify command on the restored
$ ! database to be sure that this is the case. The RMU Restore command
$ ! automatically (by default) attempts to apply all available on-disk
$ ! journal files; if you regularly back up the journal files, you must
$ ! manually apply each backed-up journal file followed by any on-disk
$ ! journal files. In this case, there is one backed-up journal file and
$ ! one on-disk journal file. Restore both the .rda and .rdb files.
$ ! The restore operation restores after-image journal file
$ ! state information for all available journal files. After restoring
$ ! JOURN_1 the operation reports that AIJ sequence numbers are
$ ! incompatible. This is expected because JOURN_1 was backed up and
$ ! has an AIJ sequence number of -1. JOURN_2 has not yet been
$ ! modified. After-image journaling was active when the database was backed
$ ! up and this message refers to JOURN_3 as being the current journal file.
```

(continued on next page)

Example 9–15 (Cont.) Restoring, Verifying, and Recovering a Database

```
$ !
$ RMU/RESTORE/NORECOVERY/LOG /ROOT=NEW$DISK:[DB.ROOTS] -
_$ RDB_BACKUPS:MF_PERSONNEL_FULL.RBF
%RMU-I-REXTXT_04, Thread 1 uses devices NEW$DISK:
%RMU-I-AIJRSTBEG, restoring after-image journal "state" information
%RMU-I-AIJRSTJRN, restoring journal "JOURN_1" information
%RMU-I-AIJRSTINC, after-image journal sequence numbers are incompatible
%RMU-I-AIJRSTDEL, journal "JOURN_1" filename
"DISK11$:[DB.JOURNAL]JOURN_1.AIJ;1" has been removed
%RMU-I-AIJRSTJRN, restoring journal "JOURN_2" information
%RMU-I-AIJRSTNMD, journal has not yet been modified
%RMU-I-AIJRSTSUC, journal "JOURN_2" successfully restored from file
"DISK12$:[DB.JOURNAL]JOURN_2.AIJ;1"
%RMU-I-AIJRSTEND, after-image journal "state" restoration complete
%RMU-I-REXTXT_00, Restored root file DB_DISK:MF_PERSONNEL.RDB;1
.
.
.
%RMU-I-AIJWASON, AIJ journaling was active when the database was backed up
%RMU-I-AIJRECFUL, Recovery of the entire database starts with AIJ file
sequence 0
%RMU-I-AIJRECBEG, recovering after-image journal "state" information
%RMU-I-AIJRSTAVL, 1 after-image journal available for use
%RMU-I-LOGMODSTR, activated after-image journal "JOURN_2"
%RMU-I-LOGMODFLG, enabled after-image journaling
%RMU-W-DOFULLBCK, full database backup should be done to ensure future
recovery
%RMU-I-AIJRECEND, after-image journal "state" recovery complete
$
$ ! The restore operation messages indicate that the
$ ! JOURN_1 is now the current journal and journaling is enabled.
$ !
$ RMU/VERIFY/ALL/LOG NEW$DISK:[DB.ROOTS]MF_PERSONNEL
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-BGNVCONST, beginning verification of constraints for database
%RMU-I-ENDVCONST, completed verification of constraints for database
%RMU-I-DBBOUND, bound to database "DB_DISK:MF_PERSONNEL.RDB;1"
%RMU-I-OPENAREA, opened storage area RDB$SYSTEM for protected retrieval
%RMU-I-BGNAIPVER, beginning AIP pages verification
%RMU-I-ENDAIPVER, completed AIP pages verification
```

(continued on next page)

Example 9–15 (Cont.) Restoring, Verifying, and Recovering a Database

```
%RMU-I-OPENAREA, opened storage area EMPIDS_OVER for protected retrieval
%RMU-I-OPENAREA, opened storage area EMPIDS_MID for protected retrieval
%RMU-I-OPENAREA, opened storage area EMPIDS_LOW for protected retrieval
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-I-ENDABMSPM, completed ABM pages verification
.
.
%RMU-S-ENDVERIFY, elapsed time for verification :      0 01:40:32.03
$ ! Applying journal files:
$ !
$ ! Use the RMU Recover command to roll forward the database changes
$ ! contained in the backed-up .aij file, BKUP_JOURN_1, which is the
$ ! system logical name for DISK21$:[DB.BKUP_JOURNAL]BKUP_JOURN_1.AIJ.
$ ! Next apply the on-disk journal file JOURN_2, which is the system
$ ! logical name for DISK12$:[DB.JOURNAL]JOURN_2.AIJ.
$ ! Refer to the new location of the database root (.rdb) file
$ ! as NEW$DISK:[DB.ROOTS]MF_PERSONNEL.rdb.
$ ! The journal files must now be applied in their proper order.
$ ! Journal files can be specified as a list, but be sure they
$ ! are in the exact order in which they need to be applied.
$ ! In this case, BKUP_JOURN_1 must be applied first; it is AIJ
$ ! sequence 0. It is the backed-up JOURN_1 file.
$ ! Recall, that JOURN_2 was not modified, it was AIJ sequence 1
$ ! and is ignored for the purposes of recovery. The recovery
$ ! operation knows that JOURN_3 must be applied next because
$ ! it is AIJ sequence 2.
$ !
$ RMU/RECOVER/TRACE /ROOT=NEW$DISK:[DB.ROOTS]MF_PERSONNEL.RDB "BKUP_JOURN_1"
%RMU-I-LOGRECSTAT, transaction with TSN 88 ignored
%RMU-I-RESTART, restarted recovery after ignoring 1 committed transaction
%RMU-I-LOGRECSTAT, transaction with TSN 96 committed
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-LOGRECSTAT, transaction with TSN 112 committed
%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-AIJVNOSYNC, AIJ file DISK12:[DB.JOURNAL]JOURN_2.AIJ;1
synchronized with database
%RMU-I-AIJSUCCEES, database recovery completed successfully
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence
number needed will be 2
```

(continued on next page)

Example 9–15 (Cont.) Restoring, Verifying, and Recovering a Database

```
$
$ RMU/RECOVER/TRACE /ROOT=NEW$DISK:[DB.ROOTS]MF_PERSONNEL.RDB "JOURN_3"
%RMU-I-LOGRECSTAT, transaction with TSN 128 committed
%RMU-I-LOGRECSTAT, transaction with TSN 129 committed
%RMU-I-AIJONEDONE, AIJ file sequence 2 roll-forward operations completed
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-AIJVNOSYNC, AIJ file DISK12$:[DB.JOURNAL]JOURN_2.AIJ;1
synchronized with database
%RMU-I-AIJSUCCESS, database recovery completed successfully
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence
number needed will be 3
$ !
$ ! If you need to apply more journal files to the same
$ ! database, you can also repeat the steps from the label
$ ! 'Applying journal files' and specify the full file
$ ! specification, including the version number of the
$ ! .aij files to be applied to the new database. An
$ ! informational message displays indicating the next
$ ! rollforward sequence number of the next .aij file
$ ! needed in the recovery process.
$ !
$ ! After recovering the database, verify it, and then do
$ ! a full database backup operation.
```

See the *Oracle RMU Reference Manual* for more examples of restoring and recovering databases.

During the database recovery, temporary files are written to your default directory. These files are deleted when the recovery operation completes.

The default file protection for the after-image journal file permits the system to read and write and the owner to read (S:RW,O:R). You can preserve this protection by using the RMU Backup After_Journal command to save copies of the journal files on tape or a disk structure.

Once you have fully restored and recovered the database, verify it, and if verification is successful, you should create a backup copy of the database by using the RMU Backup command after you enable after-image journaling again. This way you ensure that if any update transactions change the state of the database after the backup operation begins, the journal file records that activity.

Recovery-Unit Journaling and Recovery

When you start using your database, Oracle Rdb automatically begins logging a recovery-unit journal for each user's update transactions. The recovery-unit journal is sometimes called a before-image journal because it records an image of the data before the transaction makes changes to it.

If a user enters an SQL ROLLBACK statement or if a user's transaction is terminated abnormally, Oracle Rdb uses the recovery-unit journal file to roll back an update transaction.

See Chapter 9 for information about after-image journaling.

10.1 The Recovery-Unit Journal File

The recovery-unit journal keeps a record of any changes to the database definitions or the data itself. When the user decides to roll back a transaction, Oracle Rdb uses the entries in the recovery-unit journal to undo changes that were written to the database. When a system or software failure occurs and database access ceases, Oracle Rdb uses all recovery-unit journal files to complete the recovery when the database comes back on line.

Because there are as many recovery-unit journal files as there are active users of the database, Oracle Rdb must be able to access all recovery-unit journal files before the recovery is marked as complete. If a single recovery-unit journal file cannot be accessed, the database is corrupt and Oracle Rdb returns an error message.

The recovery process signals an error if the database experiences an incomplete automatic recovery, using the recovery-unit journal files. Incomplete recovery of this type can result when Oracle Rdb cannot locate *all* recovery-unit journal files associated with this database.

Recovery-unit journaling is automatic and is not controlled by the user. Oracle Rdb applies all recovery-unit journal entries whenever they are needed after a system failure. For example, when the node to which you are logged in fails and comes back on line, Oracle Rdb automatically applies all recovery-unit

journal files to the database from a different node before you can access the database again.

10.1.1 Directory Location

After you install Oracle Rdb, it automatically creates a recovery-unit journal file for each user the first time the user updates the database.

Table 10–1 describes how Oracle Rdb determines the default location for recovery-unit journal files.

Table 10–1 Default Location for Recovery-Unit Journal Files

Operating System	Default Location
Digital UNIX	<p>The database directory</p> <p>The recovery-unit journal files created in this directory are owned by the individual users. You can change the default location by defining the RDB_RUJ configuration parameter in the rdb.conf file.</p> <p>For example, assume user Jones creates an mf_personnel database that resides in the database directory /usr/jones/mf_personnel.rdb. The first time that Jones updates the personnel database, Oracle Rdb creates a recovery-unit journal (.ruj) file in the same directory. The first entry in the following directory example shows the recovery-unit journal:</p> <pre>\$ ls -l /usr/jones/mf_personnel.rdb -rwx----- . . . rdb_system\$000163682534.ruj¹ -rwx----- . . . rdb_system.rdb -rwx----- . . . rdb_system.snp</pre>

¹Oracle Rdb names recovery-unit journal files using the database name and a timestamp-generated number. This naming convention prevents multiple versions of recovery-unit journal files from using the same file name.

(continued on next page)

Table 10–1 (Cont.) Default Location for Recovery-Unit Journal Files

Operating System	Default Location
OpenVMS	<p>A top-level directory called RDMSRUJ.DIR</p> <p>Oracle Rdb creates the [RDMSRUJ] directory automatically the first time a user updates the database. The [RDMSRUJ] directory is located on the user's login (SYSSLOGIN) device and is owned by the [SYSTEM] identifier.</p> <p>Note: Do not use the SYSTEM account to perform the first database update. This is because the login (SYSSLOGIN) information for the SYSTEM user account points to a local directory (SYSSSYSROOT) on an individual node and the SYSTEM account might be defined differently on each node. Therefore, the Oracle Rdb monitor process on one node (for example, NODE_A) might not be able to access the RDMSRUJ directory on any other node. This results in an inaccessible database.</p> <p>You can change the default location of the recovery-unit journal by:</p> <ul style="list-style-type: none">• Defining the RDMSRUJ logical name in the LNM\$SYSTEM logical name table• Creating the RDMSRUJ directories using the DCL command CREATE/DIRECTORY <p>For example, assume that user Smith's top-level directory is DISK1:[SMITH], and user Jones' top-level directory is DISK2:[JONES]. If user SMITH is the first updater of any database on the system, then Oracle Rdb creates a DISK1:[RDMSRUJ] directory. Similarly, if user JONES is the first updater of the same or a different database, then Oracle Rdb creates a DISK2:[RDMSRUJ] directory. Once created, these directories do not change. For example:</p> <pre>\$ DIRECTORY DISK1:[RDM\$RUJ] MF_PERSONNEL\$009733A005D2E400.RUJ;1¹ MF_PERSONNEL\$009884CC221ADB80.RUJ;1</pre>

¹Oracle Rdb names recovery-unit journal files using the database name and a timestamp-generated number. This naming convention prevents multiple versions of recovery-unit journal files from using the same file name.

Oracle Rdb automatically creates recovery-unit journal files in a central, top-level directory to:

- Avoid problems with recovery-unit journal files scattered in various users' directories
- Prevent recovery-unit journal files from being deleted accidentally

Because Oracle Rdb must be able to locate all recovery-unit journal files in the event of a failure, you must ensure that the directories where Oracle Rdb creates recovery-unit journal files is on a commonly accessible disk.

The Oracle Rdb monitor process on any given node must be able to locate all recovery-unit journal files on every other node in the cluster to perform recovery operations. For example, when your database operates in a VMSccluster environment, you must ensure that *all* devices that contain recovery-unit journal files are served to all nodes in the cluster.

The [RDMSRUJ] directory is located on the user's login (SYS\$LOGIN) device and is owned by the [SYSTEM] identifier. To create this [RDMSRUJ] top-level directory, the [SYSTEM] identifier must have write (W) and execute (E) privileges. You can create access control lists (ACLs) on the RDMSRUJ directory; Oracle Rdb controls privileges that users need to create or delete recovery-unit journal files. The recovery-unit journal files created in this directory are owned by the individual users. The following exceptions apply to this rule:

- If the user does not have the proper access privileges to determine the user identification code of the directory, Oracle Rdb creates the recovery-unit journal file with the UIC of the user.
- If the recovery-unit journal file is created in the common recovery-unit journal directory (*device-name*:[RDMSRUJ]), the recovery-unit journal file always has the UIC of the directory.



When you use a common directory for recovery-unit journal files and multiple users are performing database update activity, you might experience disk input/output (I/O) contention on the device that contains the common directory. Because Oracle Rdb must write before-image versions of all updated rows before any transaction terminates, this I/O activity can exceed the device's capabilities to service all I/O requests.

When this happens, the I/O requests are kept in order (queued) until they can be serviced. Such a state can slow database performance. Therefore, if heavy concurrent database access is likely, you can decide about the best location for each user's recovery-unit journal file. You can also set up a systemwide login procedure that determines the recovery-unit journal location for each user logging in to the system. In this way, you can distribute I/O operations across several disk structures and ensure good database performance during heavy use.

See the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information.

10.1.2 Recovery for Update Transactions

A typical Oracle Rdb transaction (INSERT, UPDATE, DELETE) is performed between an SQL SET TRANSACTION statement or the first executable SQL statement, and a COMMIT or ROLLBACK statement. When a transaction executes a ROLLBACK statement, Oracle Rdb rolls back the incomplete transaction within the context of the recovery unit's process, replacing the before-images in that process' recovery-unit journal file into the database.

An update transaction that does not terminate with a COMMIT or ROLLBACK statement is considered to terminate abnormally. Abnormal termination occurs as a result of many possible events. The following table describes many of the events, whether the transaction is rolled back or committed, and whether Oracle Rdb creates a recovery process as a result of the event:

Event	Commit or Rollback	Recovery?
Standard SQL image rundown	COMMIT	No
DCL Ctrl/Y EXIT command (where the user enters Ctrl/Y followed by the DCL EXIT command)	ROLLBACK	No
DCL Ctrl/Y STOP command (where the user enters Ctrl/Y followed by the DCL STOP command)	ROLLBACK	Yes
DCL STOP/ID command	ROLLBACK	Yes
Oracle RMU Close command with the Abort=Delprc qualifier	ROLLBACK	Yes
Oracle RMU Close command with the Abort=Forcex qualifier	ROLLBACK	Yes
Fatal program error	ROLLBACK	Yes
Hardware or operating system failure	ROLLBACK	Yes
QUIT command in interactive SQL	ROLLBACK	Yes
EXIT command in interactive SQL	Commit	†

†See the *Oracle Rdb7 SQL Reference Manual* for more information.

For the situations in which Oracle Rdb creates recovery processes, the recovery process immediately rolls back the transaction. In cases of complete system failure, the Oracle Rdb monitor process creates one detached database recovery (DBR) process for each incomplete update transaction on the computer system. The database root file keeps track of these detached DBR processes. Oracle Rdb initiates the recovery procedure the next time the database is accessed after the system starts up.

OpenVMS OpenVMS
VAX Alpha

In a cluster with multiple nodes accessing the same database, recovery processes are initiated on a surviving node. Thus, database activity continues in a cluster despite loss of a single node. ♦

10.2 Improving Performance of the Automatic Recovery Process

If there is a system or process failure, Oracle Rdb initiates an automatic recovery process that uses the recovery-unit journal file. The number of database recovery (DBR) buffers is determined from the largest of any of the following values:

- RDMSBIND_BUFFERS logical name
- NUMBER OF BUFFERS IS argument (SQL or RDO)
- NUMBER OF RECOVERY BUFFERS IS argument (SQL or RDO)

For most failure scenarios, this strategy provides the DBR process with the same number of recovery buffers as the process being recovered. It also allows the DBA to specify a larger value, if desired.

Note

If a node failure causes Oracle Rdb to initiate the DBR process on a node other than the one where the process is failing, the DBR process cannot access the working set extent (WSEXTENT) quota from the failing process. This is because the process (on the failing node) no longer exists. Because the DBR process is unable to inherit the failing process' WSEXTENT value, it defaults to 8192 pages (databases running an Oracle Rdb software release prior to Oracle Rdb Version 7.0 default to 512 pages).

10.2.1 Setting the Number of Database Buffers

OpenVMS OpenVMS
VAX Alpha

The default value for the NUMBER OF RECOVERY BUFFERS IS parameter is 40 buffers. To allocate a specific number of database buffers, specify a value using the NUMBER OF RECOVERY BUFFERS IS option on any of these SQL statements:

- SQL CREATE DATABASE
- SQL ALTER DATABASE
- SQL IMPORT

See the full syntax diagrams for these statements in the *Oracle Rdb7 SQL Reference Manual*.

If you have a large, multfile database and you are working on a system with a large amount of memory, Oracle recommends that you specify between 100 and 200 buffers for the NUMBER OF RECOVERY BUFFERS IS option. This range provides adequate recovery in most circumstances. However, you should experiment to determine the optimal number of buffers for your particular application.

The working set extent parameter, WSEXTENT, must be large enough to hold all these buffers in memory in addition to other things mapped when the database is invoked. A high WSEXTENT value results in faster recovery time. However, if the number of buffers is too large for the specified WSEXTENT value, the system may be forced to perform virtual paging of the buffer pool. Slow performance time results because the operating system must perform virtual paging of the buffer pool in addition to reading database pages.

Displaying the Current Number of Recovery Buffers

Use the RMU Dump Header command to display the contents of the database file header, including the current value for the NUMBER OF RECOVERY BUFFERS parameter.

Displaying the Current Value of WSEXTENT

To check the value of the WSEXTENT parameter for the recovery process, run the OpenVMS Authorize (AUTHORIZE) utility and specify the SHOW command and the identifier for the recovery process.

Check the WSEXTENT value and use the SET command to increase it, if necessary.

Displaying and Setting Other Working Set Parameters

For a user process, you can use the DCL command, SHOW WORKING_SET, to display the current working set parameters, WSDEFAULT, WSQUOTA, and WSEXTENT. Then, use the DCL command, SET WORKING_SET, to adjust values for these three parameters for the user's process.

10.2.2 Adjusting the Number of Recovery Buffers

When you select a value for the NUMBER OF RECOVERY BUFFERS IS option that is between 100 to 200 buffers, and your buffer size is 12 pages to handle storage areas with page sizes of 2, 3, and 4 blocks per page in your multfile database, then the total number of pages needed for just the number of recovery buffers is between 1200 and 2400 pages.

In this case, be sure the WSEXTENT value is larger than 1200 to 2400 pages to ensure an adequate recovery time for your system. Use the NUMBER OF RECOVERY BUFFERS IS option to increase the number of buffers allocated to the recovery process, as shown in Example 10-1.

Example 10–1 Changing the Number of Recovery Buffers Allocated to the Recovery Process

```
SQL> ALTER DATABASE FILENAME 'DISK2:[USER.TEST]MF_PERSONNEL'  
cont> NUMBER OF RECOVERY BUFFERS IS 150;
```



10.3 Displaying the Contents of an .ruj File

In Example 10–3, EMPLOYEE_ID 165 (Terry Smith's) row is deleted from the EMPLOYEES table. Before the operation is committed, you can display the contents of the recovery-unit journal file by using the Recovery_Journal qualifier on the RMU Dump command, as shown in Example 10–2. Note, however, that the recovery-unit journal file is created and is empty because the results of deleting this row have not been written back to disk yet; the updates are still in the buffer.

In Example 10–3:

- VBN refers to the virtual block number in which the recovery-unit journal file entries are placed
- Sequence number indicates the order in which journal entries are generated
- JFA refers to the journal file address
- TSN refers to the transaction sequence number

To force Oracle Rdb to write the results of this deletion to the recovery-unit journal file, you can either:

- Perform additional update operations
- Preset the following logical name or configuration parameter to a small number, such as 2, and perform the operation again:
 - RDM\$BIND_BUFFERS logical name
 - RDB_BIND_BUFFERS configuration parameter

Either action forces database pages to be written to disk as other database pages are requested for updating.

Example 10–2 Displaying the Contents of an Empty Recovery-Unit Journal File

```
$ RMU/DUMP/RECOVERY_JOURNAL DUAL:[RDM$RUJ]MF_PERSONNEL$0097232208D5D240.RUJ
*-----
* Oracle Rdb V7.0-0                               7-SEP-1995 11:09:11.45
*
* Dump of Recovery Unit Journal
*   Filename: DUAL:[RDM$RUJ]MF_PERSONNEL$0097232208D5D240.RUJ;1
*
*-----
-----
          VBN 1 of 102
-----
-----
Recovery unit journal file sanity check passed ("RUJ_FILE")
Database rootfile is DUAL:[ORION]MF_PERSONNEL.RDB;1
Journal file was created by process 20211574:1
```

In Example 10–3, the recovery-unit journal file contains some results of deleting EMPLOYEE_ID 165 or Terry Smith’s row. When the row was deleted, it was also removed from all the indexes for which the EMPLOYEE_ID column was used as the column on which the index was defined.

Example 10-3 Displaying the Contents of a Recovery-Unit Journal File

```
$ RMU/DUMP/RECOVERY_JOURNAL DUAL:[RDM$RUJ]MF_PERSONNEL$0097232208D5D240.RUJ
*-----*
* Oracle Rdb V7.0-0                               7-JUN-1996 17:36:01.07
*
* Dump of Recovery Unit Journal
*   Filename: DUAL:[RDM$RUJ]MF_PERSONNEL$0097232208D5D240.RUJ;1
*
*-----*
-----
VBN 1 of 102
-----
Recovery unit journal file sanity check passed ("RUJ_FILE")
Database rootfile is DUAL:[ORION]MF_PERSONNEL.RDB;1
Journal file was created by process 20211574:1
-----
VBN 2 of 102, Sequence number 1, Transaction number 544, AIJ checkpoint TSN 251
-----
+-----+
| This JFA 2:0           Record sequence number 1
| Prior JFA 0:0         Previous TSN was 16
| Modified segment 63:2:1 with length of 76 bytes
+-----+
          001A 0000 line 1: record type 26
          00 0001 0002 1 byte in 0 sets/dynamic items
          .... 71 bytes of static data
0420886874696D53353631303000010B 0005 data '...00165Smith. .'
6E655420303231440D20847972726554 0015 data 'Terry. .D120 Ten'
75726F636F684307209F2E7244207962 0025 data 'by Dr. .Chocoru'
4932C0004D3731383330484E12208B61 0035 data 'a. .NH03817M.À2I'
          00F032006B0E77 0045 data 'w.k.2.'
+-----+
| This JFA 2:116        Record sequence number 2
| Prior JFA 2:0         Previous TSN was 64
| Modified segment 77:3:5 with length of 30 bytes
+-----+
          0020 0000 line 5: record type 32
          00 0001 0002 1 byte in 0 sets/dynamic items
          .... 25 bytes of static data
414207B4455441423536313030000113 0005 data '...00165BATE.BA'
          E000208A7374724120 0015 data ' Arts. .à'
```

(continued on next page)

Example 10–3 (Cont.) Displaying the Contents of a Recovery-Unit Journal File

```

+-----+
| This JFA 2:174                Record sequence number 3
| Prior JFA 2:116              Previous TSN was 72
| Modified segment 50:770:1 with length of 208 bytes
+-----+
... total B-tree node size: 208
002A 2004 0000 line 1: index node for set 42
0000 FFFFFFFF FFFF 0004 owner 0:-1:-1
0070 000C 112 bytes of DBKs
004D 00000003 0005 0010 duplicate record 77:3:5
004D 00000004 0005 0018 duplicate record 77:4:5
004D 00000004 000C 0020 duplicate record 77:4:12
004D 00000005 0007 0028 duplicate record 77:5:7
004D 00000005 000B 0030 duplicate record 77:5:11
004D 00000005 000C 0038 duplicate record 77:5:12
004D 00000006 000D 0040 duplicate record 77:6:13
004D 00000006 000E 0048 duplicate record 77:6:14
004D 00000006 0012 0050 duplicate record 77:6:18
004D 00000007 0008 0058 duplicate record 77:7:8
004D 00000008 0001 0060 duplicate record 77:8:1
004D 00000008 0008 0068 duplicate record 77:8:8
004D 00000008 0009 0070 duplicate record 77:8:9
004D 00000008 000A 0078 duplicate record 77:8:10
00000000000000000000000000000000 0080 unused '.....'
::: (3 duplicate lines)
FFFF FFFFFFFF FFFF 00C0 next node: -1:-1:-1
0000000000000000 00C8 MBZ '.....'
-----
VBN 3 of 102, Sequence number 4, Transaction number 544, AIJ checkpoint TSN 251
-----
... Data continued from previous block ...

```

(continued on next page)

Example 10–3 (Cont.) Displaying the Contents of a Recovery-Unit Journal File

```
+-----+
| This JFA 2:410                Record sequence number 4
| Prior JFA 2:174              Previous TSN was 72
| Modified segment 50:774:0 with length of 430 bytes
+-----+
      .... total B-tree node size: 430
      0032 2003 0000 line 0: index node for set 50
0000 FFFFFFFF FFFF 0004 owner 0:-1:-1
      00FF 000C 255 bytes of entries
      8200 000E level 1, full suffix
      00 06 0010 6 bytes stored, 0 byte prefix
      343631303000 0012 key '.00164'
FFCE 00000306 0001 0018 duplicate node 50:774:1
      05 01 001C 1 byte stored, 5 byte prefix
      3631303000 pfx '.0016'
      35 001E key '5'
      4D46 60 001F pointer 77:3:5
      05 01 0022 1 byte stored, 5 byte prefix
      3631303000 pfx '.0016'
      36 0024 key '6'
.
.
.
```

Displaying Root Files, Storage Areas, and Snapshot Files

Understanding how Oracle Rdb stores records in the database can help you maintain and fine-tune a database to better suit your applications. This chapter shows you how to use the RMU Dump command to display and interpret the contents of the database files shown in the following table:

File Type	File Extension
Root files	.rdb (For single-file databases, this file includes storage areas)
Storage area files	.rda
Snapshot files	.snp

The discussions in this chapter assume that you already understand basic database concepts about data storage (such as single-file and multifile databases) and database elements (such as tables, indexes, and domains). You might find it helpful to refer to the *Oracle Rdb7 Guide to Database Design and Definition* for background information.

See also Chapter 9 for information about using the RMU Dump command to display information about after-image journal (.aij) and recovery-unit journal (.ruj) files.

11.1 Using the RMU Dump Command

Using the RMU Dump command and the qualifiers shown in the following table, you can display internal page formats for database storage areas and snapshot files:

Command	Description
RMU Dump Area	Displays pages from data storage files and lets you examine pages in data storage areas.
RMU Dump Larea	Displays pages from one or more logical areas in data storage files. (A logical area is a group of pages assigned to a particular table or index.) Use this command to examine pages allocated to a table.
RMU Dump Snapshots	Displays pages from the snapshot file.

Refer to the *Oracle RMU Reference Manual* for information about the privileges you need to use the RMU Dump command with the Areas, Lareas, or Snapshots qualifiers.

The following example shows the RMU Dump Area command for a Digital UNIX system:

```
$ rmu -dump -areas=* mf_personnel
```

The command includes the Areas=* qualifier to display all storage areas for the mf_personnel database. The RMU Dump command potentially can produce many printed pages of output information. You should use the qualifiers and options available to customize the dump output and obtain only the information that you need.

Example 11–1 shows the output format of the RMU Dump command:

Example 11–1 Common Display Format for Logical Areas, Snapshot Files, and Database Pages

```

      ①      ②      ③
      001A 0390 line 1: record type 26
00 0001 0392 Control information
      .... 71 bytes of static data
0420886874696D53353631303000010B 0395 data '...00165Smith. .'
6E655420303231440D20847972726554 03A5 data 'Terry. .D120 Ten'
75726F636F684307209F2E7244207962 03B5 data 'by Dr.. .Chocoru'
4932C0004D3731383330484E12208B61 03C5 data 'a. .NH03817M.À2I'
      00F032006B0E77 03D5 data 'w.k.2.'
```

The format shown in Example 11–1 is common to dumps for logical areas, snapshot files, and database pages (regardless of whether the file is of mixed or uniform page format). The hexadecimal output in Example 11–1 is read from right to left:

- ① Hexadecimal notation as it appears on disk
- ② Offset of each line of the hexadecimal code

- ③ ASCII translation of the hexadecimal code (nonprinting characters are displayed as “.”).

The *Oracle RMU Reference Manual* describes the RMU Dump command and qualifiers in more detail.

11.2 Data Structures

The following table describes the data structures that manage storage areas. These data structures reside on pages in the storage area:

Data Structure	Description
ABM	An area bit map (ABM) structure located on ABM pages. Each ABM entry on the ABM page identifies a SPAM page that is assigned page clumps for a particular logical area.
AIP	An area inventory page (AIP) structure located on AIP pages. Each AIP entry identifies ABM pages associated with a particular logical area. The AIP is pointed to by the database root file.
SPAM	A space area management (SPAM) structure located on SPAM pages. Each SPAM entry on the SPAM page identifies the assignment of logical areas (tables and indexes) to clumps of pages. The SPAM data structure is common to both the uniform page format storage area and the mixed page format storage area.

For single-file and multifile databases with uniform page format storage areas, SPAM pages, ABM pages, and AIP pages control the placement and retrieval of the data pages on which data rows and index records are stored. See Figure 11–1 for more information.

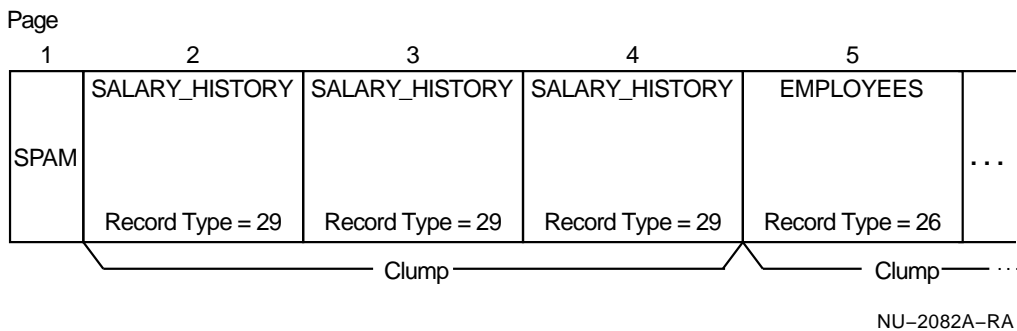
For multifile databases with mixed page format storage areas, only SPAM pages control the placement and retrieval of data pages. See Section 11.2.2 for more information.

Snapshot (.snp) files contain only the data page type; snapshot files do not have SPAM pages, ABM pages, or AIP pages. See Section 11.5 for more information.

11.2.1 Storage Areas with Uniform Page Format

When you create a storage area with uniform page format, the storage area file is divided into groups of n pages, called **clumps**, where n is equal to the buffer size divided by the page size. Both buffer size and page size are user-specified values. By default, the buffer size is 6 blocks and the page size is 1024 bytes or 2 blocks long, resulting in clumps of 3 pages, as shown in Figure 11–1.

Figure 11–1 Storage Area with Uniform Page Format



Each clump is assigned to a specific logical area. (A logical area refers to the way Oracle Rdb internally partitions the logical entities that comprise the database). Each logical area corresponds to either a database management function, a data row storage area, or an index segment storage area. All pages in a specific logical area contain records or index nodes from the same table identified by its record type ID number.

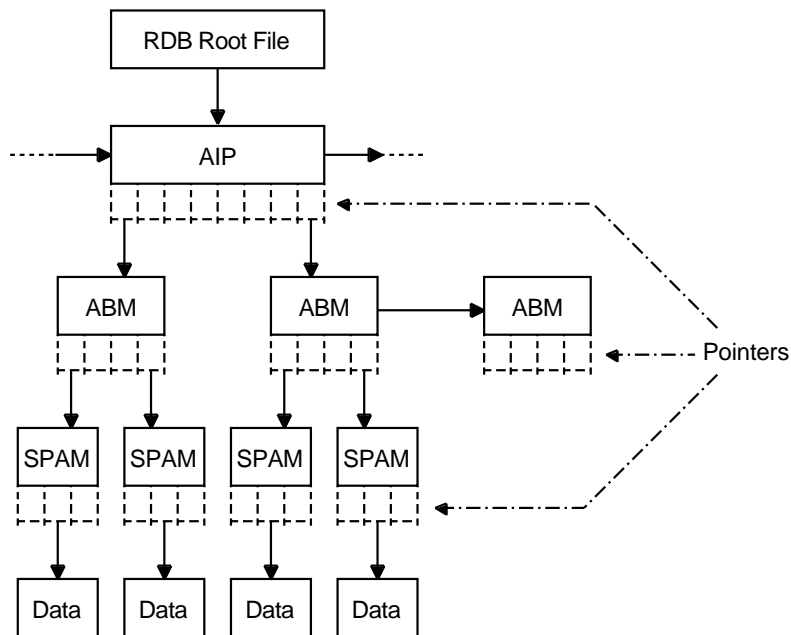
For single-file databases:

- All indexes for the same table in the RDB\$SYSTEM area share the same logical area
- Rows from different tables cannot occur on different pages within the same clump because these rows represent different logical areas
- Each clump of three pages is described by the record type ID number, as shown in Figure 11–1
- For tables, pages are never mixed at the clump level
- Index node records from different index structures, for the same table, can occur on different pages within the same clump

If Oracle Rdb needs more space to store a row in a table and there are pages available in the database that are not allocated to any logical areas, a clump of database pages is reserved and allocated for a logical area, one clump at a time. If, in the process of trying to allocate a clump of pages to a logical area, there is no more space available, the physical area is extended. Only one clump is allocated to the table at a time.

Areas with uniform page format are managed by the SPAM, ABM, and AIP data structures (described in Section 11.2). Figure 11–2 shows the relationship between the database root file (.rdb) file, the AIP and ABM data structures, and the SPAM and data pages.

Figure 11–2 Mapping AIP and ABM Data Structures to SPAM and Data Pages



Legend

-
- ABM = Area bit map page
 - AIP = Area inventory page
 - Data = Data pages containing rows of the EMPLOYEES table
 - SPAM = Space area management page

ZK-1005A-GE

The SPAM data structure is common to both the uniform page format storage area and the mixed page format storage area. The SPAM page for storage areas with uniform page format indicates the fullness threshold for each of the data pages and contains a list of clumps and the logical areas to which they belong. To scan a logical area, Oracle Rdb takes the following steps:

1. Locates the AIP entry for the area that locates the ABM chain for the logical area
2. Examines the ABM chain and locates the SPAM pages that have page clump assignments for the logical area

3. Examines the SPAM pages and locates the page clumps that belong to the logical area
 4. Examines each data page in the page clump that belongs to the logical area
- See Chapter 13 for more information on SPAM pages, ABM data structures, and AIP data structures.

11.2.2 Storage Areas with Mixed Page Format

A storage area with a mixed page format contains pages that can hold rows from more than one table or hashed-index structure and nodes for sorted indexes. Mixed page format:

- Allows related rows from different tables to be clustered together (shown in Figure 11–3) and identified by their record type ID numbers
- Allows index records to be clustered together with table rows on the same page and identified in the system record by their set-type (logical area) ID numbers

The table row database key (dbkey) pointer found in the index structure associates the index with the table or tables to which it belongs, (see Example 12–13 and the *Oracle Rdb7 Guide to Database Design and Definition* for examples of this type of row and index clustering).

Figure 11–3 Storage Area with Mixed Page Format

Page	1	2	3
SPAM	EMPLOYEES Row (Record Type=26) SALARY_HISTORY Row (Record Type=29) SALARY_HISTORY Row (Record Type=29) SALARY_HISTORY Row (Record Type=29) ⋮ System Record	EMPLOYEES Row (Record Type=26) SALARY_HISTORY Row (Record Type=29) SALARY_HISTORY Row (Record Type=29) SALARY_HISTORY Row (Record Type=29) ⋮ System Record	...

NU-2083A-RA

Data pages do not “belong” to a particular kind of clump; thus, SPAM pages for storage areas that are mixed page format contain only fullness threshold information. SPAM pages do not contain any clump management information.

A mixed page format storage area contains a file of database data pages, any of which can store row occurrences. Rows are stored in a mixed page format storage area either by checking the SPAM page first for a data page with space to store the complete record or by using a hashed index to hash the row to a target data page (without checking the SPAM page).

A placement index is an index (usually a hashed index) that is specified in the PLACEMENT VIA INDEX option in a storage map definition. If the complete record can fit on the target data page, it is stored there along with the associated index structures (system record, hash bucket, and duplicate hash node [if needed], if it is a hashed index). If the complete record cannot fit on the target data page, the following happens depending on whether a placement index is used to place the row:

- When no placement index is used, the SPAM page is checked to locate available space on another data page within the SPAM interval of data pages in the storage area.
- If a placement index is used to place the row in the storage area, data pages in the buffer are checked first for available space. Only when no available space is found on the data pages in the buffer will the SPAM page be checked to locate another data page with available space.

If table rows use a placement index, then even if the record does not fit on a particular target data page, the record is likely to be stored within the same buffer load of data pages, particularly if the storage area is initially sized correctly. If the data pages in the buffer are full and no available space is located from the SPAM page, Oracle Rdb begins a sequential analysis of the remaining SPAM pages in the storage area if SPAM pages are enabled. The sequential search continues to the end of the storage area and wraps around to the beginning of the storage area. Once the first SPAM interval is accessed again and no available space is found, the storage area is extended and the row is placed on a data page in the extended portion of the file.

11.3 Displaying Data Storage Files

Use the RMU Dump Area command to display a range of pages from the data storage file or files:

- In a single-file database, there is one data storage file, the .rdb file
- In a multifile database, the .rdb file contains information about all data storage areas (.rda files), and there may be separate .rda files that contain data from one or more tables

Examples 11–2 and 11–3 show how to display all storage areas or selected storage areas of the mf_personnel sample database.

Displaying All Storage Areas in the Database

Example 11–2 shows the data storage areas of the multfile database mf_personnel. Example 11–2 displays the initial parts of two storage areas:

- **RDB\$SYSTEM**

The uniform page format storage area, RDB\$SYSTEM, shows the first SPAM page header followed by SPAM entries that indicate data page capacities (3=full, 0=empty), then the clump count, and then the logical area designation for each clump that indicates the logical area to which it and its data pages belong for all logical areas in the database.

- **EMPIDS_LOW**

The mixed page format storage area, EMPIDS_LOW, displays the first SPAM page header for this physical area, then the SPAM entries for page fullness thresholds for the data storage pages as shown in Example 11–2.

This type of information displays for all physical areas of the database when you enter the RMU Dump Area command.

Example 11–2 Storage Areas of the mf_personnel Database

```
$ RMU/DUMP/AREA/START=1/END=1 MF_PERSONNEL
*-----
* Oracle Rdb V7.0-0                               14-FEB-1996 09:34:02.85
*
* Dump of Live area RDB$SYSTEM
*   Filename: $DUAL:[ORION]MF_PERS_DEFAULT.RDA;1
*   Database: $DUAL:[ORION]MF_PERSONNEL.RDB;1
*-----
          0001 00000001 0000 page 1, physical area 1 (space mgmt) ❶
          BD045E16 0006 checksum = BD045E16
      80000000 00000001 000A Fast incremental backup TSN = 1
          0000 0001 0012 1 free byte, 0 locked
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 0016 pages 2-65: threshold 3 ❷
FFF3CFFFFFFFFFFFFFFFFFFFFFFFFF 0026 pages 66-119: threshold 3
```

(continued on next page)

Example 11–2 (Cont.) Storage Areas of the mf_personnel Database

```

      .
      .
      .
      016B 0127 363 clumps
      4001 0129 pages 2-4, logical area 16385
      0001 012B pages 5-7, logical area 1
      0002 012D pages 8-10, logical area 2
      .
      .
      .
-----
* Oracle Rdb V7.0-0
*                               14-FEB-1996 09:39:50.74
*
* Dump of Live area EMPIDS_LOW
*   Filename: $DUAL:[ORION]EMPIDS_LOW.RDA;1
*   Database: $DUAL:[ORION]MF_PERSONNEL.RDB;1
*
-----
      0002 00000001 0000 page 1, physical area 2 (space mgmt)
      12E68E03 0006 checksum = 12E68E03
      80000000 00000010 000A Fast incremental backup TSN = 16
      0000 03B4 0012 948 free bytes, 0 locked
00000000000000000000000010003200D000C02 0016 page 2: threshold 2
      pages 3-6: threshold 0
      page 7: threshold 3
      .
      .
      .

```

LEGEND

- ① SPAM page header
- ② SPAM entries indicating page capacities as follows:
 - 3=95% to 100% full
 - 2=85% to less than 95% full
 - 1=70% to less than 85% full
 - 0=0% to less than 70% full
- ③ Clump count and logical area entry for a clump of data pages

Displaying Specific Storage Areas in the Database

To see a specific storage area, use the RMU Dump Area command and specify the name of the storage area, as shown in Example 11–3. In this example of the mixed page format storage area, EMP_INFO, the SPAM page header is followed by SPAM entries for page fullness thresholds for data storage pages.

Example 11-3 EMP_INFO Storage Area

```

$ RMU/DUMP/AREA=EMP_INFO MF_PERSONNEL
*-----
* Oracle Rdb V7.0-0                               14-FEB-1996 10:20:23.29
*
* Dump of Live area EMP_INFO
*   Filename: $DUAL:[ORION]EMP_INFO.RDA;1
*   Database: $DUAL:[ORION]MF_PERSONNEL.RDB;1
*-----
          0008 00000001 0000 page 1, physical area 8 (space mgmt) ❶
                03BA800A 0006 checksum = 03BA800A
      80000000 00000007 000A Fast incremental backup TSN = 7
                0000 03AA 0012 938 free bytes, 0 locked

00000000000000000000000000000000FFFF 0016 pages 2-9: threshold 3 ❷
                                                pages 10-65: threshold 0
00000000000000000000000000000000 0026 pages 66-129: threshold 0
00000000000000000000000000000000 0036 pages 130-193: threshold 0
                000000000000 0046 pages 194-217: threshold 0

00000000000000000000000000000000 004C MBZ free '.....'
                ::: (58 duplicate lines)
      00000000000000000000 03FC MBZ free '.....'

          0008 00000002 0000 page 2, physical area 8 ❸
                FD66380F 0006 checksum = FD66380F
      009714FB 982F69E0 000A time stamp = 20-JAN-1996 16:00:30.59
                0000 0036 0012 54 free bytes, 0 locked
                0012 0016 17 lines
                .
                .
          0039 01AA 0048 line 12: offset 01AA, 57 bytes ❹
          0039 0170 004C line 13: offset 0170, 57 bytes
          0032 013E 0050 line 14: offset 013E, 50 bytes
          0035 0108 0054 line 15: offset 0108, 53 bytes
          0032 00D6 0058 line 16: offset 00D6, 50 bytes

          00000000 005C line 0: TSN 0 ❺
          00000007 0060 line 1: TSN 7
          00000007 0064 line 2: TSN 7
          00000007 0068 line 3: TSN 7

```

(continued on next page)

Example 11-3 (Cont.) EMP_INFO Storage Area

```

.
.
00000000000000000000000000000000 00A8 free space '.....' ⑥
      001F 00D6 line 16: record type 31 ⑦
      00 0001 00D8 Control information
      .... 45 bytes of static data
746973726576696E5520454D55000118 00DB data '...UME Universit'
6F724F042085656E69614D20666F2079 00EB data 'y of Maine. .Oro'
      E03337343430454D07208E6F6E 00FB data 'no. .ME04473à'
      001F 0108 line 15: record type 31
      00 0001 010A Control information
      .... 48 bytes of static data
2064726F666E6174534E415453000118 010D data '...STANStanford '
61745307208579746973726576696E55 011D data 'University. .Sta'
E03530333439414307208B64726F666E 012D data 'nford. .CA94305à'
      00 013D padding '.'
.
.
      0008 0000001F 0000 page 31, physical area 8 ③
      184626AC 0006 checksum = 184626AC
009714FB 0208FF80 000A time stamp = 20-JAN-1996 15:56:18.68
      0000 03C4 0012 964 free bytes, 0 locked
      0001 0016 1 line
      0005 03E4 0018 line 0: offset 03E4, 5 bytes ④
      00000000 001C line 0: TSN 0 ⑤
00000000000000000000000000000000 0020 free space '.....' ⑥
      ::: (59 duplicate lines)
      00000000 03E0 free space '....'
      2001 03E4 line 0: SYSTEM record ⑧
      00 0001 03E6 1 byte in 0 sets/dynamic items
0000000000 03E9 padding '.....'
      FFFFFFFF 03EE snap page pointer -1 ⑨
      00000000 03F2 snap pointer TSN 0
      0000 03F6 MBZ '.....'
      00000000 03F8 page sequence number 0
      00000000 03FC MBZ '.....'

```

(continued on next page)

Example 11–3 (Cont.) EMP_INFO Storage Area

LEGEND

- ❶ SPAM page header
- ❷ SPAM entries indicating page capacities as follows:
 - 3=95% to 100% full
 - 2=85% to less than 95% full
 - 1=70% to less than 85% full
 - 0=0% to less than 70% full
- ❸ Data page header
- ❹ Line index
- ❺ TSN index
- ❻ Free space
- ❼ User-stored data storage record
- ❽ System record
- ❾ Page tail

See Chapter 12 for more information about each of the seven components of a data page. Refer to the threshold setting to determine the fullness thresholds.

11.4 Displaying Logical Areas

The RMU Dump Larea command displays the storage area pages contained in one or more logical areas (a group of pages assigned to a particular table or index). Each table stored in an Oracle Rdb database has its own logical area. In a single-file database, each table in the database is stored in its own logical area. System tables, user-defined tables, and indexes for a table are also stored in the logical area.

You can use the RMU Dump Larea command to display pages for storage areas with the uniform page format only. For storage areas with a uniform page format, ABM pages track the location of SPAM pages, which keep track of information on the data pages.

The following table describes how the RMU Dump Larea command handles uniform page format and mixed page format storage areas:

For . . .	The RMU Dump Larea command . . .
A single-file database or a multifile database that has storage areas with a uniform page format	Displays the ABM page of the logical area first.

For ...	The RMU Dump Larea command ...
A multifile database that has storage areas with a mixed page format	Does not display logical areas because a storage area with the mixed page format does not contain ABM pages. The RMU Dump Larea command returns the database header and a message.

See Section 11.2 for a description of the ABM and SPAM page structures.

Displaying a Logical Area in the RDB\$SYSTEM Storage Area

Example 11–4 shows an abbreviated view of the ABM pages in the CANDIDATES logical area contained in the RDB\$SYSTEM storage area (which always has a uniform page format).

Example 11–4 CANDIDATES Logical Area

```

$ RMU/DUMP/LAREA=CANDIDATES MF_PERSONNEL
*-----
* Oracle Rdb V7.0-0                                14-FEB-1996 10:45:50.12
*
* Dump of Logical area CANDIDATES
*   Filename: $DUAL:[ORION]MF_PERS_DEFAULT.RDA;1
*   Database: $DUAL:[ORION]MF_PERSONNEL.RDB;1
*-----
          ①
0001 00002D2 0000 page 722, physical area 1           ①
          7B5D6255 0006 checksum = 7B5D6255
009714FB 5EEAC4E0 000A time stamp = 20-JAN-1996 15:58:54.51
          0000 0004 0012 4 free bytes, 0 locked
          000002D3 0016 next area bit map page 723
          00000000 001A max set bit index 0
          00000000 001E MBZ '....'
          00001E60 0022 bitvector count 7776
00000000000000000000000000000000000000000001 0026 bitvector '.....'
00000000000000000000000000000000000000000000 0036 bitvector '.....'
          :::: (58 duplicate lines)
00000000000000000000000000000000000000000000 03E6 bitvector '.....'
          00000000 03F2 MBZ '....'
          804A 03F6 bitmap page for logical area 74     ②
          00000000 03F8 page sequence number 0
          00000000 03FC MBZ '....'

```

(continued on next page)

Example 11–4 (Cont.) CANDIDATES Logical Area

```

0001 000002D3 0000 page 723, physical area 1
      2BDD6244 0006 checksum = 2BDD6244
009714FB 5ED87560 000A time stamp = 20-JAN-1996 15:58:54.39
      0000 0004 0012 4 free bytes, 0 locked

      000002D4 0016 next area bit map page 724
      00000000 001A max set bit index 0
      00000000 001E MBZ '....'
      00001E60 0022 bitvector count 7776
000000000000000000000000000000000000000000000000 0026 bitvector '.....'
      :::: (59 duplicate lines)
000000000000000000000000000000000000000000000000 03E6 bitvector '.....'

      00000000 03F2 MBZ '....'

      804A 03F6 bitmap page for logical area 74
      00000000 03F8 page sequence number 0
      00000000 03FC MBZ '....'
      .
      .
      .

```

LEGEND

- ❶ Data page header
- ❷ Page tail

In Example 11–4, the RMU Dump Larea command displays all database pages that belong to the CANDIDATES table. You can use the RMU Dump Larea command to look at rows for a particular table. For example:

If ...	Then ...
The rows are stored on pages that are close together	The rows can be read from disk into database buffers in a few I/O operations. Queries that require scans of an entire table benefit both from rows being stored on pages that are close together and from large buffers that you define in SQL CREATE and ALTER statements.
Your application is update intensive with many users making updates	Rows should not be placed close together.
Your application is update intensive with the updates performed by a single user	The rows are likely to be placed close together.

See Section 13.3.1 for a description of the components of the ABM page type.

Obtaining a List of All Logical Areas

Enter the Oracle RMU commands shown in Example 11–5 to obtain a list of all logical areas. The commands in the example display only header information for each logical area in the `mf_personnel` database, including the logical area name.

Example 11–5 Commands to Display the First Page of All Logical Areas

```
$ RMU/DUMP/LAREA/START=1/END=1 MF_PERSONNEL
$
$ RMU/DUMP/LAREA/START=2000000000 MF_PERSONNEL
```

Displaying Logical Area ID Information

To display logical area ID information, use the RMU Analyze command shown in Example 11–6.

Example 11–6 Command to Display Area IDs of All Logical Areas

```
$ RMU/ANALYZE/OPTIONS=DEBUG/END=1 MF_PERSONNEL
```

Refer to the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information about using the RMU Analyze command and interpreting information in the display output.

Displaying Logical Area Numbers and Names

Also, you can display logical area numbers and names by dumping the `RDB$AIP` logical area, which maintains a list of all logical areas in the database. Example 11–7 shows two of the logical areas (entries #0 and #1) in the `mf_personnel` database. This information is useful for finding out the names of the logical areas that contain indexes for tables. The command shown in Example 11–7, entered on a Digital UNIX system, encloses the name `RDB$AIP` in single quotation marks. The quotation marks are unnecessary when you enter the command to dump the `RDB$AIP` logical area on an OpenVMS system.

Example 11-7 Logical Area RDB\$AIP Listing Logical Area Numbers and Names of the mf_ personnel Database

```

$ rmu -dump -larea='RDB$AIP' mf_personnel.rdb
*-----*
* Oracle Rdb V7.0-00                               13-FEB-1996 15:29:50.73
*
* Dump of Logical area RDB$AIP
*   Filename: $DUAL:[ORION]MF_PERS_DEFAULT.RDA;1
*   Database: $DUAL:[ORION]MF_PERSONNEL.RDB;1
*-----*
      0001 00000002 0000 page 2, physical area 1           ❶
          EBB7795E 0006 checksum = EBB7795E
009714FB 05FCA7E0 000A time stamp = 20-JAN-1996 15:56:25.31
          0000 0022 0012 34 free bytes, 0 locked
      00000003 0016 next area inventory page 3           ❷
000000000000000000 001A MBZ '.....'
          0010 0022 16 logical area entries           ❸
                    entry #0
      00000005 0024 first area bitmap page 5           ❹
0001 0001 0028 logical area 1, physical area 1           ❺
          15 002C area name length 21 bytes           ❻
54535F44445544E454D47455324424452 002D area name 'RDB$SEGMENTED_ST'
00000000000000000000000053474E4952 003D area name 'RINGS.....'
      00000001 004C snaps enabled TSN 1           ❼
          00A2 0050 record length 162 bytes           ❽
00000000 0052 MBZ '....'
          01 0056 entry is in use           ❾
          0000 0057 MBZ '..'
          000000 0059 thresholds are (0,0,0)           ❿
          000000 005C MBZ '....'
          .
          .
          .
      4001 03F6 logical area 16385
00000000 03F8 page sequence number 0
00000000 03FC MBZ '....'

```

(continued on next page)

Example 11–7 (Cont.) Logical Area RDB\$AIP Listing Logical Area Numbers and Names of the mf_personnel Database

LEGEND

- ❶ AIP header
- ❷ Pointer to next AIP
- ❸ Number of logical area entries on the AIP
- ❹ Pointer to first ABM page for this logical area
- ❺ Logical area and storage area numbers
- ❻ Logical area name
- ❼ Last transaction to perform an exclusive update to this logical area
- ❽ Original record length
- ❾ Used for setting SPAM thresholds
- ❿ User-specified logical area SPAM thresholds

The entry number gives the number of each entry in the RDB\$AIP logical area for the current page. The information stored for each entry includes the logical area number and logical area name for each logical area in the database.

Error When Displaying Logical Area for a Mixed Page Format

In a multfile database, Oracle RMU cannot display information about logical areas that reside in a mixed page format storage area. Example 11–8 shows the message that is returned instead of the logical area display.

Example 11–8 Attempts to Display the Contents of the EMPLOYEES Logical Area

```
$ RMU/DUMP/LAREA=EMPLOYEES MF_PERSONNEL
*-----
* Oracle Rdb V7.0-0                               14-FEB-1996 11:00:29.45
*
* Dump of Logical area EMPLOYEES
*   Filename: $DUAL:[ORION]EMPIDS_LOW.RDA;1
*   Database: $DUAL:[ORION]MF_PERSONNEL.RDB;1
*-----
```

This data resides in a MIXED FORMAT storage area. It may not be dumped by LOGICAL AREA.

Using the Area and Larea Qualifiers Together

You can use the Lareas and Area qualifiers together to obtain more specific information. The following command displays the pages, if any, of a logical area called CANDIDATES in the default RDB\$SYSTEM storage area:

```
$ RMU/DUMP/AREA=RDB$SYSTEM/LAREA=CANDIDATES
```

11.5 Displaying Snapshot Files

The RMU Dump Snapshots command displays the contents of your snapshot (.snp) file, letting you see exactly what rows and index nodes are stored there. Of particular interest might be the number of versions of a given row in the snapshot file, indicating long read-only transactions concurrent with heavy update activity on a table. Displaying the contents of your snapshot file gives you an idea of how much you are using that file.

The RMU Dump Snapshots command shown in Example 11–9 works similarly to the RMU Dump Area command.

Example 11–9 First Page of Selected Snapshot Files

```
$ RMU/DUMP/SNAPSHOTS/START=1/END=1 MF_PERSONNEL
*-----*
* Oracle Rdb V7.0-0                               14-FEB-1996 11:04:29.16
*
* Dump of Snapshot area for live area RDB$SYSTEM
*   Filename: DUAL:[ORION.TEST]MF_PERS_DEFAULT.SNP;1
*   Database: DUAL:[ORION.TEST]MF_PERSONNEL.RDB;1
*
*-----*
          4001 00000001 0000 page 1, physical area 16385           ❶
          F1C15BDC 0006 checksum = F1C15BDC
00972322 1328A4C0 000A time stamp = 7-JAN-1996 16:11:14.06
          0000 02F4 0012 756 free bytes, 0 locked
          0001 0016 1 line
          00D0 0316 0018 line 0: offset 0316, 208 bytes           ❷
          00000048 001C line 0: TSN 72                             ❸
          0001 0020 line 0 -> live line: 1                         ❹
0000000000000000000000000000000000000000000000000000000 0022 free space '.....'
          :::: (46 duplicate lines)
          000000 0312 free space '...'                             ❺
```

(continued on next page)

Example 11-9 (Cont.) First Page of Selected Snapshot Files

```

        .... total B-tree node size:208
        0032 2004 0316 line 1: index node for set 50
0000 FFFFFFFF FFFF 031A owner 0:-1:-1
        0070 0322 112 bytes of DBKs
004D 00000003 0005 0326 duplicate record 77:3:5
004D 00000004 0005 032E duplicate record 77:4:5
004D 00000004 000C 0336 duplicate record 77:4:12
004D 00000005 0007 033E duplicate record 77:5:7
004D 00000005 000B 0346 duplicate record 77:5:11
004D 00000005 000C 034E duplicate record 77:5:12
004D 00000006 000D 0356 duplicate record 77:6:13
004D 00000006 000E 035E duplicate record 77:6:14
004D 00000006 0012 0366 duplicate record 77:6:18
004D 00000007 0008 036E duplicate record 77:7:8
004D 00000008 0001 0376 duplicate record 77:8:1
004D 00000008 0008 037E duplicate record 77:8:8
004D 00000008 0009 0386 duplicate record 77:8:9
004D 00000008 000A 038E duplicate record 77:8:10
00000000000000000000000000000000 0396 unused '.....'
        ::: (3 duplicate lines)
        FFFF FFFFFFFF FFFF 03D6 next node: -1:-1:-1
        0000000000000000 03DE MBZ '.....'

        00000302 03E6 live page pointer 770
        00000220 03EA max TSN 544
        FFFFFFFF 03EE snap page pointer -1
        00000000 03F2 snap pointer TSN 0
        0000 03F6 MBZ '..'
        00000000 03F8 page sequence number 0
        00000000 03FC MBZ '.....'

```

```

*-----*
* Oracle Rdb V7.0-0                               14-FEB-1996 11:04:30.12
*
* Dump of Snapshot area for live area EMPIDS_LOW
*   Filename: DUAL:[ORION.TEST]EMPIDS_LOW.SNP;1
*   Database: DUAL:[ORION.TEST]MF_PERSONNEL.RDB;1
*-----*

```

(continued on next page)

Example 11–9 (Cont.) First Page of Selected Snapshot Files

```

4002 00000001 0000 page 1, physical area 16386
      BAC60353 0006 checksum = BAC60353
00972322 13B99A20 000A time stamp = 7-JAN-1996 16:11:15.01
      0000 01CA 0012 458 free bytes, 0 locked
      0008 0016 8 lines
      004C 039A 0018 line 0: offset 039A, 76 bytes
      0025 0375 001C line 1: offset 0375, 37 bytes
      0033 0342 0020 line 2: offset 0342, 51 bytes
      005C 02E6 0024 line 3: offset 02E6, 92 bytes
      002B 02BB 0028 line 4: offset 02BB, 43 bytes
      002B 0290 002C line 5: offset 0290, 43 bytes
      002B 0265 0030 line 6: offset 0265, 43 bytes
      0033 0232 0034 line 7: offset 0232, 51 bytes

      00000010 0038 line 0: TSN 16
      00000028 003C line 1: TSN 40
      00000028 0040 line 2: TSN 40
      00000028 0044 line 3: TSN 40
      00000028 0048 line 4: TSN 40
      00000028 004C line 5: TSN 40
      00000028 0050 line 6: TSN 40
      00000010 0054 line 7: TSN 16

      0001 0058 line 0 -> live line: 1
      0009 005A line 1 -> live line: 9
      0005 005C line 2 -> live line: 5
      0007 005E line 3 -> live line: 7
      0008 0060 line 4 -> live line: 8
      0006 0062 line 5 -> live line: 6
      0004 0064 line 6 -> live line: 4
      0002 0066 line 7 -> live line: 2

0000000000000000000000000000000000000000000000000000000 0068 free space '.....'
      ::: (27 duplicate lines)
      0000 0228 free space '..'

      003A 2005 0232 line 2: bucket for hash index 58
      .... total hash bucket size: 51
FFFF FFFFFFFF FFFF 0236 bucket overflow -1:-1:-1
      00 023E flags 0
      00000001 023F duplicate count 1
003F 00000002 0001 0243 pointer 63:2:1
      06 024B key len: 6 bytes
      353631303000 024C key: '.00165'
      00000001 0252 duplicate count 1
003F 00000002 0003 0256 pointer 63:2:3
      06 025E key len: 6 bytes
      303931303000 025F key: '.00190'

```

(continued on next page)

Example 11–9 (Cont.) First Page of Selected Snapshot Files

```

          001D 0265 line 4: record type 29
          00 0001 0267 Control information
          .... 38 bytes of static data
CEFB80004B4353413536313030000124 026A data '$..00165ASCK..ûî'
4E52485000853002AFCC80000082BE82 027A data '.....Ï.0..PHRN'
          C03130323030 028A data '00201À'
          .
          .
          .
00000002 03E6 live page pointer 2
00000220 03EA max TSN 544
FFFFFFFF 03EE snap page pointer -1
00000000 03F2 snap pointer TSN 0
          0000 03F6 MBZ '..'
00000000 03F8 page sequence number 0
          .
          .
          .

```

LEGEND

- ❶ Snap page header
- ❷ Line index
- ❸ TSN index
- ❹ Live line index
- ❺ Free space
- ❻ Record snapshot
- ❼ Page tail

Snapshot file pages contain many of the same components as data storage pages. The following components are added to help manage snapshot file transactions:

- ❹ Live line index

A third-line index that relates each line on the snapshot file (snap) page with the corresponding line on the live page. A live page is a data page, and is distinguished from a snapshot file page.
- ❼ Page tail

A page tail that points to the live page corresponding to this snapshot page and the most recent transaction represented on this snapshot page. The two fields, live page pointer, and max TSN are specific to snapshot files.

In addition, notice that the area number component in the snapshot page header is different from the live data page header:

❶ Snap page header

The area number written on each page in a snapshot file is the same as the one written in the corresponding live storage area (.rda) file, but incremented by 16384. This is the number that the RMU Dump command displays. It is the decimal representation of the same area number as the following example shows:

Live Area Number		Snapshot Area Number	
HEX	DECIMAL	HEX	DECIMAL
0001	1	4001	16385
0002	2	4002	16386

In cases when a majority of the line indexes are empty and the majority of the transaction sequence numbers (TSNs) are zero, there are no snapshots of records on the pages displayed in the MF_PERS_DEFAULT.SNP and MF_PERS_SEGSTR.SNP snapshot files. Example 11–11 shows a snapshot file page containing snapshots of several rows.

The RMU Dump Snapshots command is useful if you know which pages in the snapshot file or files you want to see. You can use the Output qualifier to direct the display information into a text file, as shown in Example 11–10.

Example 11–10 Displaying Selected Pages of the EMPIDS_LOW Snapshot File to an Output File

```
$ RMU/DUMP/SHOTS=(EMPIDS_LOW)/START=25/END=35/OUTPUT=SNAP_DUMP.LIS -
_ $ MF_PERSONNEL
```

The command in Example 11–10 displays pages 25 through 35 of the EMPIDS_LOW snapshot file into a file called SNAP_DUMP.LIS that you can edit or print. Snapshot files contain only one type of database page format, a snapshot file page.

11.5.1 Snapshot Page Tail

Example 11–11 is another example of a snapshot file.

Example 11–11 Snapshot File Showing the Page Tail

4004 00000001	0000	page 1, physical area 16388	❶
F1C15BDC	0006	checksum = F1C15BDC	
00972322 1328A4C0	000A	time stamp = 7-JAN-1996 16:11:14.06	
0000 0362	0012	866 free bytes, 0 locked	
0003	0016	3 lines	
0012 03D4	0018	line 0: offset 03D4, 18 bytes	❷
001E 03B6	001C	line 1: offset 03B6, 30 bytes	
001E 0398	0020	line 2: offset 0398, 30 bytes	
000001A1	0024	line 0: TSN 417	❸
000001D8	0028	line 1: TSN 472	
000001D8	002C	line 2: TSN 472	
000C	0030	line 0 -> live line: 12	❹
000E	0032	line 1 -> live line: 14	
0010	0034	line 2 -> live line: 16	
00000000000000000000000000000000	0036	free space '.....'	❺
		::: (53 duplicate lines)	
0000	0396	free space '..'	
000D	0398	line 16: record type 13	❻
00 0001	039A	Control information	
	25 bytes of static data	
530220896D616853333037303000010A	039D	data '...00703Sham. .S'	
F8FF0100D720866D61	03AD	data 'am. x...ø'	
000D	03B6	line 14: record type 13	❼
00 0001	03B8	Control information	
	25 bytes of static data	
530220896D616853323037303000010A	03BB	data '...00702Sham. .S'	
F8FF0100D720866D61	03CB	data 'am. x...ø'	
000D	03D4	line 12: record type 13	❼
00 0001	03D6	Control information	
	13 bytes of static data	
FEFF0100EF3130373030000106	03D9	data '...00701i...p'	
00000153	03E6	live page pointer 339	❼
0000022B	03EA	max TSN 555	❽
FFFFFFFF	03EE	snap page pointer -1	❾
00000000	03F2	snap pointer TSN 0	❿
0000	03F6	MBZ '..'	
00000000	03F8	page sequence number 0	⓫
00000000	03FC	MBZ '....'	

(continued on next page)

Example 11–11 (Cont.) Snapshot File Showing the Page Tail

LEGEND

- ❶ Snap page header
- ❷ Line index
- ❸ TSN index
- ❹ Live line index
- ❺ Free space
- ❻ User-stored data storage record
- ❼ Live page pointer
- ❽ Maximum TSN
- ❾ Snap page pointer
- ❿ Snap pointer TSN
- ⓫ Page sequence number

As shown in Example 11–11, the snapshot file page tail contains 26 bytes (not 18, as found on a data page) because of two additional fields:

- The live page pointer
- The maximum transaction sequence number (TSN) or last snap TSN

In addition, the snap page pointer and the snap pointer TSN have slightly different meanings on a snapshot file page.

The individual fields are as follows:

❼ Live page pointer

This 4-byte field (00000153) shows the corresponding live data page for this .snp file page.

❽ Maximum TSN

This 4-byte field (0000022B) shows the most recent transaction to write to this snapshot file page. For every active read/write transaction (except read/write exclusive and batch-update transactions), data rows are written to a page in the snapshot file by the transaction that is doing the updating. This value represents the highest or current TSN value for the snapshot file page.

❾ Snap page pointer

This 4-byte field (FFFFFFFF) shows the next snapshot file page in the snapshot chain. In this case, there is no next snapshot file page.

⑩ Snap pointer TSN

This 4-byte field (00000000) shows the most recent transaction to write to the next snapshot file page. This value and the max TSN value together determine the span of transactions that wrote to this snapshot page.

⑪ Page sequence number

This 4-byte field (00000000) shows the page sequence number. The page sequence number is used for recovery purposes for databases running Oracle Rdb V4.1 and higher.

12

Displaying the Contents of Data Storage Pages

Data storage pages consist of one or more disk blocks of 512 bytes each. The number of blocks in a page is defined when you create a database.

This chapter describes the data storage page and its components that are listed in Table 12–1:

Table 12–1 Components of a Data Storage Page

Component	Description	Reference
Page header	Fixed-length portion of the page that contains page and storage area information.	Section 12.1
Line index	Directory to all the storage segments on the page.	Section 12.2
TSN index	An index of transaction sequence number (TSN) entries; one for each storage segment or storage record on the page.	Section 12.3
Free space	Space that is allocated to the storage segments during database transactions.	Section 12.4
Locked free space	Free space that is associated (using the transaction identification number (TID)) with a transaction when a user attaches to the database. Unlocked free space is not associated with any transactions.	Section 12.4
Storage segments (storage records)	The largest portion of the data storage page that stores either a whole storage record or fragmented segments. A record becomes fragmented when it is too large to fit on a single database page.	Section 12.5

(continued on next page)

Table 12–1 (Cont.) Components of a Data Storage Page

Component	Description	Reference
Page tail	Fixed format (18 byte) portion of the data storage page that follows the storage segments at the end of the data page.	Section 12.6

Many of these components are also components of a snapshot (.snp) file page. Figure 12–1 shows the components of a data storage page.

Figure 12–1 Data Storage Page Components in a Mixed Page Format Storage Area

Bytes Locked Space	Bytes Free Space	Timestamp	Page Checksum	Storage Area Identification Number	Page Number	Page Header	
...			Length of Storage Segment	Offset from Page Beginning to Storage Segment	Line Index Count	Line Index	
TSN Index					TSN of Lines	TSN Index	
Locked Space						Locked Space	
Free Space						Free Space	
Storage Segments User-Stored and Index Nodes						Storage Segments	
System Storage Record (Mixed Storage Area Only)							
			Logical Area	Maximum Snap TSN	Snap Page Pointer	Page Sequence Number	Page Tail

NU-2084A-RA

Note that only the mixed page format storage areas contain one system record per page.

12.1 Page Header for a Data Storage Page

The page header is a fixed-length portion of the page that contains page and storage area information, as shown in Example 12–1.

Example 12–1 Page Header for a Data Storage Page

```
    ②    ①  
0001 00000060 0000 page 96, physical area 1  
    ③  
    00000000 0006 checksum = 00000000  
④  
0089943A DFF43F20 000A time stamp = 4-AUG-1987 15:53:59.57  
    ⑥    ⑤  
    0000 031C 0012 796 free bytes, 0 locked
```

The page header shown in Example 12–1 contains the following hexadecimal information as read from right to left and from top to bottom:

① Page number

Each page is identified in its storage area by the page number field (00000060). This number is a permanent part of the database page.

② Storage area number

This number identifies the storage area (0001) in which the page is contained. In combination with the page number, the storage area number identifies a page in the database. Like the page number, the storage area number is a permanent part of the page.

③ Page checksum

The page checksum (00000000) indicates valid data on the bit level, allowing Oracle Rdb to detect invalid transfers of data.

④ Timestamp of last modification

The timestamp (0089943A DFF43F20) is changed each time a page is written back to the database. This value shows the time and date of the last modification to that page.

The time is stored with 100 nanosecond precision but is displayed to only a hundredth-of-a-second precision in the ASCII translation. The accuracy is system dependent.

⑤ Number of bytes of free space

If a page is not filled with data and the associated overhead, there is free space (031C) on that page. This page header entry shows how much free space is left on that page.

⑥ Number of bytes of locked free space

There is also an entry for locked free space (0000), free space allocated to transaction activity until it is finished with that page.

12.2 Line Index for a Data Storage Page

The line index is a dynamic portion of a database page. It is a directory to all the storage segments on the page. The line index begins with a 2-byte field that contains a count of the number of line index entries.

Following the line index count is a variable number of line index entries, one for every storage segment (storage record) in the page. Each line index entry contains a pair of 2-byte fields—an offset address relative to the beginning of the page and length value that indicates the size of the storage segment. Oracle Rdb uses the offset address and length to locate the data on the page, given the page and line number components of the database key (dbkey).

Example 12–2 shows four line index entries.

Example 12–2 Line Index for a Data Storage Page

```

  ①
0004 0016 4 lines
④  ③  ②
000D 03F2 0018 line 0: offset 03F2, 13 bytes
003A 03B8 001C line 1: offset 03B8, 58 bytes
003A 037E 0020 line 2: offset 037E, 58 bytes
003A 0344 0024 line 3: offset 0344, 58 bytes
```

The line index shown in Example 12–2 contains the following:

① Line index count

This field (0004) shows the number of line index entries on that page.

② Line index entries

There is a line index entry (line 0 through line 3) for each storage segment on the page.

③ Offset from beginning of page to storage segment

The offset address (03F2), combined with the length entries, identifies the location of a storage segment on the page. The location of a specific storage segment on a page is the sum of the page address and the offset address.

This location can be represented as:

$$\text{storage - segment - address} = \text{page - address} + \text{offsetaddress}$$

④ Length of storage segment (000D)

The size in bytes (000D) of the storage segment.

12.3 TSN Index for a Data Storage Page

Transaction sequence number (TSN) entries follow the final line index entry and precede the free space entry. There is one entry for each storage segment or storage record on the page. Each record has a TSN assigned to it to keep track of the last transaction that updated it. The sequence of TSN index entries corresponds to the sequence of line index entries. For example, the third line index entry and the third TSN index entry are paired and refer to the same storage segment.

When a new storage segment is added to a page, a line index entry is added. All TSN entries are moved to make room for the line index, and a new TSN entry is added. In Example 12–3, each TSN entry consists of a 4-byte hexadecimal field (callout ① (00000000)), and identifies the transaction that last stored, modified, or erased the storage segment (TSN 22).

To find the storage segment to which a TSN entry refers, you must note its position in the TSN (callout ② (line 0)), look at the line index entry that corresponds positionally (callout ③ (line 0)), and use the offset address (callout ④ (01EA)) in the line index entry. In Example 12–3, the offset 01EA identifies the storage segment for TSN 0.

Example 12–3 shows a line index and TSN index entries. It also shows the relationship of the TSN index to the line index.

Example 12–3 TSN Index for a Data Storage Page

	0003	0016	3 lines
	④		
Line Index	0005 01EA	0018	line 0: offset 01EA, 5 bytes ③
	0078 0172	001C	line 1: offset 0172, 120 bytes
	0078 00FA	0020	line 2: offset 00FA, 120 bytes
	①		
TSN Index	00000000	0024	line 0: TSN 0 ②
	00000016	0028	line 1: TSN 22
	00000016	002C	line 2: TSN 22

12.4 Locked and Unlocked Free Space for a Data Storage Page

Free space begins after the final TSN index entry and continues to the first storage segment. The free space becomes larger when storage segments are deleted and smaller when they are added. A portion of free space can be locked by a transaction. Free space is allocated to storage segments from the bottom and to locked free space from the top. The page fills as storage segments displace free space. When the convergence of storage segment space and line index entries leaves no usable free space, the page is full.

Each byte of locked free space is associated with a specific database attach. Unlocked free space is not associated with any transaction. Oracle Rdb makes these associations by using the transaction identification number (TID). A TID is assigned when a user attaches to a database and is deassigned only when that user detaches. The TID is not related to the TSN. The TID is stored in each word of the locked free space, as shown in Example 12-4.

Free space is locked by the TID that created it, in case that transaction has to roll back, for example, when a record is deleted. When a transaction creates free space, that free space is locked by Oracle Rdb for use by only that database attach. When the transaction needs storage space, Oracle Rdb first attempts to use the locked free space allocated to that transaction. If the space is inadequate, Oracle Rdb then allocates unlocked free space to that transaction.

Locked free space is available to be claimed by other users only after the user whose transactions locked it enters an SQL DISCONNECT statement. Locked free space can be reclaimed when needed by the user who locked it. Locked line indexes, however, do not work the same way as locked free space. Locked line index entries are not used again until the user's process detaches from the database. This means that the same process that deletes a line cannot use that particular line index entry again.

Example 12-4 shows the format of the locked and unlocked free space components of a data storage page.

Example 12–4 Locked and Unlocked Free Space for a Data Storage Page

```

0001 00000154 0000 page 340, physical area 1
      12B33540 0006 checksum = 12B33540
008FF4E3 21F9DB00 000A time stamp = 12-JAN-1988 08:30:03.44
      0012 03BC 0012 956 free bytes, 18 locked ❶
      0001 0016 1 line
      ❷
0002 0000 0018 line 0: locked by 2

      00000000 001C line 0: TSN 0
000200020000200020002000200020002 0020 locked space '.....' ❸
      0002 0030 locked space '..'
00640064006400640064006400640064 0032 free space 'd.d.d.d.d.d.d.' ❹
000000000000000000000000000000640064 0042 free space 'd.d.....'
0000000000000000000000000000000000 0052 free space '.....'
      :::: (54 duplicate lines)
303000010E000001000D00000000000000 03C2 free space '.....00'
06000001000D7474656E657547383035 03D2 free space '508Guenett.....'
      FFFF0100EF31303730300001 03E2 free space '..00701i...p'
      .
      .
      .

```

The locked and unlocked free space shown in Example 12–4 contains the following components:

- ❶ The page header (offset 0000 through 0015)
Note that 18 bytes are locked (0012) and that 956 bytes are free (03BC). See the examples and descriptions for the format of the page header and line index entries in Sections 12.1 and 12.2.
- ❷ The transaction ID (TID)
Every word in locked free space is assigned to the TID of the user (0002) who locked it. In this case, 18 bytes are locked by TID 2.
- ❸ Locked space
This space begins at 0020 and ends at 0031.
- ❹ Free space
This space begins at 0032 and ends at 03E2. (The 0064 in the free space indicates that TID HEX 0064 had locked free space reserved at some stage.)

12.5 Storage Segment Structure for a Data Storage Page

Storage segments take up most of the space on a database page. A segment can be stored either whole (called a storage record) or fragmented. Fragmented records consist of a primary segment and any number of secondary segments.

Oracle Rdb fragments a record when it is too large to fit on a single database page or when a modify operation expands the record and it then becomes too large to fit in the free space on its page. Use the RMU Analyze command to determine if records have been fragmented. The RMU Dump command displays the fragmented records. Section 12.7 describes fragmented records in more detail.

Oracle Rdb stores a minimum of 10 bytes (2 bytes overhead for the record type plus 8 bytes for the dbkey pointer) for each record on the page, so there is always room on the page to convert the storage record or segment into the first primary segment of a fragmented record.

The storage area is automatically extended when there is no room for new or modified rows. The size of the extent is determined by the EXTENT IS (extension-options) parameter of the SQL CREATE DATABASE statement.

Section 12.5.1, Section 12.5.2, and Section 12.5.3 show the relationship of three different types of storage segments on a database page. These segments are as follows:

- User-stored data storage segments (records), see Section 12.5.1
- List (segmented string) storage segments (records), see Section 12.5.2
- Index node storage segments (records), see Section 12.5.3

Each storage segment on a database page has a storage record header that contains a 14-bit storage record identification number and 2-bit flags that identify the record as fragmented or whole. The format of the rest of the storage segment depends on whether the segment contains user-stored data, a list, or an index node. There is a storage segment for each occurrence of a record type.

12.5.1 User-Stored Data Storage Segments

Example 12–5 represents the information stored in a user-stored data storage segment (storage record).

Example 12–5 User-Stored Data Storage Segment

```

      ② ①
      001A 0390 line 1: record type 26
      ④ ③
      00 0001 0392 Control information
                .... 71 bytes of static data
      ⑥          ⑤
0420886874696D53353631303000010B 0395 data '...00165Smith. .'
6E655420303231440D20847972726554 03A5 data 'Terry. .D120 Ten'
75726F636F684307209F2E7244207962 03B5 data 'by Dr.. .Chocoru'
4932C0004D3731383330484E12208B61 03C5 data 'a. .NH03817M.À2I'
      ⑦
      00F032006B0E77 03D5 data 'w.k.2ð.'
```

A user-stored data storage segment shown in Example 12–5 contains the following components:

❶ The storage record type

Each storage segment is identified by its storage record (hexadecimal 1A) or number (26). This is the RDB\$RELATION_ID value that is stored in RDB\$RELATIONS system relation for this table. Oracle Rdb uses this internal number to find the storage segment. This number is determined by the order in which tables are created in the database.

❷ Frag flags

The frag flags indicate whether a segment is whole (the value of the flag is 00) or fragmented. If the record has been fragmented, the first bit indicates the presence of a following segment and the second bit, the presence of a preceding fragment. The first segment is called the primary storage segment and the others are called secondary storage segments. In the case of the primary storage segment, the frag flags would indicate 10 (following segment, no preceding segment). A secondary storage segment with a following segment would be 11. The final secondary storage segment would be 01.

The top 2 bits (00 to the left of the vertical bar) of the following word are the frag flags:

```
000D = 00 | 00 0000 0000 1101
```

❸ The total length of pointer clusters

For Oracle Rdb, the total cluster length is always set to one (0001).

④ The cluster count

For Oracle Rdb, the number of pointer clusters is always null (00).

⑤ User data

This portion of the storage record contains user-supplied data (offset 0395 through 03DB). If the table to which this row belongs has compression enabled, the first byte is the compression byte (0B in this case), followed by the 2-byte record version number (0001), and then the user data. If the table has compression disabled, then there are no compression bytes in the user-stored data storage segment.

⑥ The record version number

Oracle Rdb allows for dynamic record modification. This number determines what version (0001) of the record definition Oracle Rdb will use.

⑦ A null bit vector

The bit vector (00F0) consists of 1 bit for each column in a record. If a bit is set, the value for the corresponding column in the row is missing.

12.5.2 List Storage Segments

The SQL LIST OF BYTE VARYING data type is a special data type designed to handle large pieces of data with a segmented internal structure. Except for the length of the list segments, Oracle Rdb does not know anything about the data contained in a list. The list column in a table contains the dbkey pointer to the list but not the actual list contents. This is different from a data row that is defined as any other type of data type in which the column contains the data.

Lists use the LIST OF BYTE VARYING data type, which is implemented as a list of record segments. Oracle Rdb defines a special name to refer to the segments of a list similar to the way column names refer to columns in a database. The name denotes a value of a segment and is named RDB\$VALUE. Because the lengths of segments can vary from 0 to 64K bytes, Oracle Rdb defines a name for the length of a segment as RDB\$LENGTH.

Oracle Rdb supports the following three different list formats. See the LIST OF BYTE VARYING data type description in the *Oracle Rdb7 SQL Reference Manual* for a graphical description of the first two list formats.

- Chained format

This type of list format is used for list data stored on all devices except write-once storage areas on write-once, read-many (WORM) devices. See Example 12–6. This is the default format if you define the `RDMS$USE_OLD_SEGMENTED_STRING` logical name or the `RDB_USE_OLD_SEGMENTED_STRING` configuration parameter.

- Indexed format

The default list format for storing list data for V4.1 and higher versions of Oracle Rdb and for storing list data on write-once storage areas on WORM devices even if the `RDMS$USE_OLD_SEGMENTED_STRING` logical name or `RDB_USE_OLD_SEGMENTED_STRING` configuration parameter is defined to use the chained list format for storing lists on read/write media. See Examples 12–7, 12–8, and 12–10.

- Single-segment format

A list format supported since V1.0 of Oracle Rdb if the amount of data can fit within the list (segmented string) buffer, which is controlled by the `RDMS$BIND_SEGMENTED_STRING_BUFFER` logical name or the `RDB_USE_OLD_SEGMENTED_STRING` configuration parameter. The logical name or configuration parameter must not be defined to use the single-segment list format. See the LIST OF BYTE VARYING data type description in the *Oracle Rdb7 SQL Reference Manual* for more information. See Example 12–9.

The chained list format is comprised of a list column in a table row that contains a dbkey pointer to the first data segment of the list. Each storage segment contains 5 bytes of control information and a dbkey pointing to the next segment (8 bytes); therefore, each list data portion is limited to 65,522 bytes (65,535 bytes – 13 bytes). In addition, the first storage segment includes some special overhead to describe the entire list: total length of the entire list (a QUADWORD), the total number of segments (a LONGWORD), and the length of the longest segment (a WORD). These 14 bytes (8 + 2 + 4 bytes) of overhead are subtracted from the size of the data segment (65,522 bytes) to calculate the size of the first data storage segment (65,508 bytes). Each subsequent data storage segment is limited to 65,522 bytes. See Example 12–6.

The indexed list format is comprised of a list column in a table row containing a dbkey that points to the first segment of the list. The first segment contains the following: some overhead (12 bytes), the dbkey to the next pointer segment (8 bytes), the number of dbkeys (4 bytes), the total string length (8 bytes), the number of data segments (4 bytes), the number of pointer segments (4 bytes), the longest segment (4 bytes) (for a total of 44 bytes plus an additional 5 bytes of control information); and a directory of all the dbkey pointers (8 bytes) paired with the segment length (4 bytes) for each data segment in the list. So

the first segment contains 65,486 bytes (65,535 – 44 – 5 bytes) of remaining space to store an array of 12-byte data segment (dbkey and segment length pairs) pointers.

If this first directory segment fills to capacity, a second directory segment is created that contains a dbkey for the next pointer segment (8 bytes), number of dbkeys (4 bytes), and a continuation of the directory of the dbkey/segment length pointers (8 bytes + 4 bytes) pointing to the data segments in the list. Therefore, the second and subsequent directory segments contain 65,535 – 12 – 5 bytes or 65,518 bytes of space to store an array of 12-byte data segment (dbkey and segment length pairs) pointers. Each data segment contains 5 bytes of control information leaving 65,530 bytes (65,535 bytes – 5 bytes) of space in which to store list or data. See Examples 12–7, 12–8, and 12–10.

The single-segment list consists of a type field that is used to differentiate it from primary segments and data segments. This type of format omits the use of pointers and other overhead and allows a single I/O operation to read the segment. See Example 12–9.

Lists can be stored in more than one storage area as defined by the storage map definition for the storage areas containing these lists. For each storage area associated with a table, there is information stored that points to the actual list.

Prior to Oracle Rdb V6.0, all LIST OF BYTE VARYING (segmented string) records were written to the database as record type zero (0). This meant that the RMU Dump command could not distinguish a pure data record from a structure record.

Beginning with Oracle Rdb V6.0, list structure records are typed, allowing the RMU Dump command to format them as shown in Examples 12–6 through 12–10. You can inspect the page display to look for the pointer to the list or display the contents of the actual list for the storage areas defined to contain this type of data. For another way to display the actual contents of a list, see the sections on lists in the *Oracle Rdb7 SQL Reference Manual* and the *Oracle Rdb7 Guide to SQL Programming*.

The following list describes the five structure record types and the data segment:

1. BLOB (binary large object) primary chained segment

This first segment of the chained style lists contains statistics as well as data and a pointer to the next segment. This type of segment is referenced by the table row. See Example 12–6.

Example 12–6 Primary and Secondary Chained Segment

```

      ①
      2008 01B0 line 11: blob primary chained segment
00 0001 01B2 Control information
      .... total blob segment size: 43

      ②
0001 00000096 000A 01B5 next chained segment 1:150:10

      ③
00000000 00000033 01BD total length of segments 0;51

      ④
00000003 01C5 number of segments 3

      ⑤
0015 01C9 length of longest segment 21

      ⑥
65657268742061207369207369687409 01CB data '.this is a three'
      656E696C20 01DB data ' line'

      2009 01E0 line 10: blob secondary chained segment
00 0001 01E2 Control information
      .... total blob segment size: 20
0001 00000096 0009 01E5 next chained segment 1:150:9
6E6F69747069726373656409 01ED data '.description'
00 01F9 padding '.'

      2009 01FA line 9: blob secondary chained segment
00 0001 01FC Control information
      .... total blob segment size: 26
      FFFF FFFFFFFF FFFF 01FF next chained segment -1:-1:-1
736162617461642065687420726F6609 0207 data '.for the databas'
      2065 0217 data 'e '
00 0219 padding '.'

```

The primary and secondary chained segment shown in Example 12–6 contains the following information:

- The data area contains the dbkey of the first segment and either FFFFFFFF FFFFFFFF or 00000000 00000000, depending on whether there is a list actually stored.
- The primary and secondary chained segments contain the following hexadecimal information as read from right to left:

① The first 5 bytes

These bytes are the record type and frag flags and 3 bytes of control information.

② The next 8 bytes

These bytes are the dbkey of the next segment. If it is the last segment, then the dbkey is -1:-1:-1.

③ The next 8 bytes

These bytes provide the total length of the list and are part of the information block.

④ The next 4 bytes

These bytes provide the total number of segments in the list and are part of the information block.

⑤ The next 2 bytes

These bytes provide the length of the longest segment in the list and are part of the information block.

⑥ The actual data in the list

2. BLOB secondary chained segment

Subsequent segments of the chained style lists contain only data and a pointer to the next segment. This type of segment is only referenced by other chained segments. See Example 12–6.

3. BLOB primary indexed segment

The first segment of the indexed style list contains statistics as well as pointers to data segments. This type of segment is referenced by the table row. See Example 12–7.

Example 12–7 Primary Indexed Segment

```

    ①
      200B 002C line 0: blob primary pointer segment
00 0001 002E Control information
      .... total blob segment size: 956

    ②
00000000 0031 segment type 0
00000000 0035 MBZ '....'
00000000 0039 unused '....'
    ③
```

(continued on next page)

Example 12–7 (Cont.) Primary Indexed Segment

```
0001 000001BD 0007 003D next pointer segment 1:445:7
      ④
      0000004C 0045 number of dbkeys 76
      ⑤
00000000 0000090F 0049 total length of segments 0:2319
      ⑥
      00000055 0051 number of data segments 85
      ⑦
      00000002 0055 number of pointer segments 2
      ⑧
      0000001C 0059 length of longest segment 28
      ⑨
00000012 0001 00000096 0003 005D data segment 1:150:3 (len 18)
0000000C 0001 00000096 0004 0069 data segment 1:150:4 (len 12)
      .
      .
0000001C 0001 000001BC 0010 03D5 data segment 1:444:16 (len 28)
0000001C 0001 000001BC 0011 03E1 data segment 1:444:17 (len 28)
      00 03ED padding '.'
```

The primary indexed segment shown in Example 12–7 contains the following hexadecimal information as read from right to left:

- ① The first 5 bytes
These bytes are the record type and frag flags and 3 bytes of control information.
- ② The next 12 bytes
These bytes are the overhead bytes.
- ③ The next 8 bytes
These bytes are the dbkey of the next pointer segment. If it is the last segment, then the dbkey is -1:-1:-1.
- ④ The next 4 bytes
These bytes provide the number of dbkeys.
- ⑤ The next 8 bytes
These bytes provide the total string length.
- ⑥ The next 4 bytes
These bytes provide the number of data segments.
- ⑦ The next 4 bytes

These bytes provide the number of pointer segments.

⑧ The next 4 bytes

These bytes provide the length of the longest segment.

⑨ The next 12 bytes and subsequent groups of 12 bytes

These bytes are the directory of the dbkey/segment length pointers (8 bytes + 4 bytes) pointing to the data segments in the list.

4. BLOB secondary indexed segment

Subsequent segments of the indexed style list contain only pointers to data segments. This type of segment is only referenced by other indexed segments. See Example 12–8.

Example 12–8 Secondary Indexed Segment

```

      ①
      200C 029A line 7: blob secondary pointer segment
00 0001 029C Control information
      .... total blob segment size: 120
      ②
FFFF FFFFFFFF FFFF 029F next pointer segment -1:-1:-1
      ③
00000009 02A7 number of dbkeys 9
      ④
0000001C 0001 000001BC 0012 02AB data segment 1:444:18 (len 28)
0000001C 0001 000001BC 0013 02B7 data segment 1:444:19 (len 28)
0000001C 0001 000001BD 0000 02C3 data segment 1:445:0 (len 28)
0000001C 0001 000001BD 0001 02CF data segment 1:445:1 (len 28)
0000001C 0001 000001BD 0002 02DB data segment 1:445:2 (len 28)
0000001C 0001 000001BD 0003 02E7 data segment 1:445:3 (len 28)
0000001C 0001 000001BD 0004 02F3 data segment 1:445:4 (len 28)
0000001C 0001 000001BD 0005 02FF data segment 1:445:5 (len 28)
00000004 0001 000001BD 0006 030B data segment 1:445:6 (len 4)
      00 0317 padding '.'
```

The secondary indexed segment shown in Example 12–8 contains the following hexadecimal information as read from right to left:

① The first 5 bytes

These bytes are the record type and frag flags and 3 bytes of control information.

② The next 8 bytes

These bytes are the dbkey of the next pointer segment. If it is the last segment, then the dbkey is -1:-1:-1.

- ③ The next 4 bytes
These bytes provide the number of dbkeys.
- ④ The next 12 bytes and subsequent groups of 12 bytes
These bytes are the continuation of the directory of the dbkey/segment length pointers (8 bytes + 4 bytes) pointing to the data segments in the list.

5. BLOB simple data segment

This type of segment is referenced by the table row and contains only the data. This style of pointer segment is used when only a single segment exists for this list. See Example 12–9.

Example 12–9 Simple Data Segment

```

      ①
      200A 0146 line 19: blob simple data segment
00 0001 0148 Control information
      .... total blob segment size: 28
      ②
      FFFFFFFE 014B segment type -2
      ③
0000010C0001FFFFFF000019FF0000010C 014F data '.....'
      3FFFFFFF00000011 015F data '.....?'
      00 0167 padding '.'

```

The simple data segment shown in Example 12–9 contains the following hexadecimal information as read from right to left:

- ① The first 5 bytes
These bytes are the record type and frag flags and 3 bytes of control information.
- ② The next 4 bytes
These bytes are the segment type. In this case, -2 represents a single-segment list format.
- ③ The actual data in the list

6. Data segment

Pure data segments still remain as record type of zero (0). This type of segment is only referenced by the indexed segments. See Example 12–10.

Example 12–10 Data Segments Referenced by Primary and Secondary Indexed Segments

```

      ①
      0000 0318 line 6: record type 0
00 0001 031A Control information
      .... 4 bytes of static data
      ②
20202020 031D data '  '
      00 0321 padding '.'
      0000 0322 line 5: record type 0
00 0001 0324 Control information
      .... 28 bytes of static data
6161616161616161616161616161616109 0327 data '.aaaaaaaaaaaaaa'
      6161616161616161616161616161 0337 data 'aaaaaaaaaaaa'
      00 0343 padding '.'

```

The pure data segment shown in Example 12–10 contains the following hexadecimal information as read from right to left:

① The first 5 bytes

These bytes are the record type and frag flags and 3 bytes of control information.

② The actual data in the list

Lists written prior to V6.0 still have a record type of zero (0) and therefore remain unformatted by the RMU Dump command. This in no way affects the usage of lists by applications.

Restriction

If a list segment is fragmented, the structure information cannot be formatted by the RMU Dump command.

12.5.3 Index Node Storage Segments

Oracle Rdb allows you to define indexes on tables to more quickly retrieve records. Index nodes are database records that are used to locate data records by key value. Oracle Rdb supports two kinds of indexes—sorted and hashed indexes:

- Sorted indexes (ranked and non-ranked) can be placed in either uniform or mixed page format storage areas
- Hashed indexes can be placed only in mixed page format storage areas

The pages assigned to an index are formatted in the same manner as data pages. However, the storage segments of an index contain index nodes instead of data records.

The following table shows how the index node structure is dependent on the type of index:

Ranked sorted indexes contain the following types of nodes:

- Index key nodes

Index key nodes contain unique index keys and pointers to other nodes. A sorted index structure can have many levels of index key nodes. Level 1 nodes, also known as leaf nodes, point to rows in the table or to overflow nodes. Level 2 (and above) nodes point to lower-level index nodes.

With ranked sorted indexes, Oracle Rdb compresses duplicates using a byte-aligned bitmap compression. It compresses a list of dbkeys that point to data rows with the same index values and stores the list in the index key nodes. If you store duplicate entries, Level 1 nodes contain the compressed list of dbkeys.

- Overflow nodes

Oracle Rdb creates overflow nodes when the compressed list of duplicates in the index key node overflows that node. The overflow nodes contain a bitmap-compressed list of dbkeys and pointers to the next overflow node.

Non-ranked sorted indexes contain the following types of nodes:

- Index key nodes

Index key nodes contain unique index keys and pointers to other nodes. An index structure can have many levels of index key nodes. Level 1 nodes, also known as leaf nodes, point to rows in the table or to duplicate index rows. Level 2 (and above) nodes point to lower-level index nodes.

- Duplicate nodes

Oracle Rdb creates duplicate index nodes when you define the index without using the UNIQUE keyword and you store a duplicate key value. The duplicate nodes contain a list of the duplicate row dbkeys.

Hashed indexes contain the following three kinds of index node records:

- System records
- Hash buckets
- Duplicate node records

Because hashed indexes are restricted to storage areas with mixed page format, each page in the storage area has a system record that points to the hashed index records (hash buckets) on the page. See Section 12.5.3.2 for more information about system records. System records are not used by sorted indexes stored in a mixed page format storage area. For this reason, system records are considered part of the hashed index structure.

12.5.3.1 Sorted Index Node Records

For uniform page format storage areas, one or more sorted indexes and their structures defined for the same table are stored in clumps of pages and can be within the same clump as long as each index is defined for the same table. In a single-file database, the name of the logical area is the same as the name of the first index defined for that table. If this first index is deleted, the name of the logical area remains unchanged. For a multfile database, each sorted index has a distinct logical area name.

For each index in a table, there is a separate B-tree index structure. Each B-tree structure is created by linking index nodes together by means of dbkeys.

Because sorted index nodes are themselves records, they are also subject to record locking.

Note

If you define an index node that is larger than a database page, the node is fragmented when stored; its size is not increased or decreased. For performance reasons, index nodes larger than a database page are not recommended.

Example 12–11 shows an index node record with the index keys that make up an index record for a non-ranked sorted index. This index record contains an index key called ADMN at offset 0218 as the first index key shown in the DEPARTMENTS_INDEX sorted index.

with a following segment would be 11. The final secondary storage segment would be 01.

② Storage set ID (set)

This number (4E) identifies the index to which an index node belongs. This is the RDB\$INDEX_ID column value stored in the RDB\$INDICES system table and known as the logical area number of the index.

③ Storage record type ID

Each storage segment is identified by its storage record number (2003). Oracle Rdb uses this internal number to indicate that this segment is a B-tree node.

④ Uncompressed owner dbkey (owner)

The uncompressed owner dbkey field (004F FFFFFFFF FFFF) identifies the table or owner of the node. The logical area number (LNO) and page number (PNO) are always -1:-1.

⑤ Data region length

The number of bytes for the index scroll (00BC) or 188 bytes. The **index scroll** is the portion of the index node in which all the data contained in an index node for a particular record resides.

⑥ Level type

The level (8200) of the B-tree structure at which the index node resides. The value 8000 represents a full suffix. If there were suffix compression for the level 1 node, the value would be 200. Level 1 is the bottom level of the B-tree structure. This entry also shows if Oracle Rdb has compressed the suffix of the data values stored in a node (level 1 nodes do not have suffix compression; all other levels do).

The following items comprise the index scroll portion of the index node. The index scroll shown in Example 12-11 contains key entries, each of which includes:

⑦ Prefix length

The length of the prefix from the previous entry (00).

⑧ Key length

The length of the data in the index node entry (05).

⑨ Key data

The actual data (minus the prefix) (4E4D444100). The first entry of each index node contains a noncompressed key value. Other entries may contain compressed key values.

⑩ Compressed dbkey

The dbkey pointer to a data record, to a duplicate index node, or to the index node at the next level of the B-tree structure. In this case, this compressed dbkey (32,06) expands to a dbkey pointer (79:2:1), that points to a data row on this same page at line entry 1, but is not shown in this example.

⑪ Unused space

Space that is currently not in the scroll (offset 02D2 through 03CC).

⑫ The uncompressed dbkey pointer to the next node

The dbkey pointer (FFFF FFFFFFFF FFFF) to the next index node in the B-tree structure. The dbkey pointer -1:-1:-1 indicates that there is no other node in the B-tree structure to which to point.

Oracle Rdb automatically balances index nodes to minimize the number of levels in the tree. Balancing also tends to keep the depth of the index relatively uniform. In large indexes, each level can result in an I/O operation or disk access.

For sorted non-ranked indexes, when two or more identical keys exist in an index, that key's scroll entry points to a duplicate node instead of a data row. The duplicate index node lists the uncompressed dbkeys for each row that has the same key value. Duplicate index nodes can be chained together if there are many duplicate key values.

For sorted ranked indexes, when two or more identical keys exist in an index, that key's scroll entry points to nothing (-1:-1:-1) and the dbkeys of the duplicates are stored in compressed form in the data section of the entry. If there are enough duplicates to overflow the node, Oracle Rdb creates an overflow node and treats it as an extension of the data section.

Example 12-12 shows an index node record with the index keys that make up an index record for a ranked sorted index, SH_EMP_ID_RANK. Two index keys are shown in their entirety. The first, containing duplicates, is 00173 at offset 009D; the second, a unique key, is 00319 at offset 0298.

Example 12–12 Sorted Index Node Segment for a Ranked Sorted Index

```

$ RMU/DUMP/AREA=SALARY_HISTORY MF_PERSONNEL
.
.
.
. .... total B-tree node size: 430
0000 003D 200D 0088 line 2 (51:116:2) index: set 61
      FFFFFFFF FFFF 008C owner 0:-1:-1
      00DF 0094 223 bytes of entries
      8200 0096 level 1, full suffix
  ① ② ③ ④
40 00 06 001A 0098 6 bytes stored, 0 byte prefix
333731303000 009D key '.00173'
      0F 2D 00A3 overflow pointer -1:-1:-1 ⑤
      0008 00A5 entry cardinality 8. ⑥
      0001 00A7 leaf cardinality 0. ⑦
      5111 60 00A9 reference pointer 81:0:0 ⑧
      0009 00AC 9 byte bitmap containing 8 records ⑨
0051 00000004 0016 00AE duplicate record 81:4:22
0051 00000004 0017 00AE duplicate record 81:4:23
0051 00000005 0001 00B3 duplicate record 81:5:1
0051 00000005 0002 00B3 duplicate record 81:5:2
0051 00000005 0003 00B3 duplicate record 81:5:3
0051 00000005 0004 00B3 duplicate record 81:5:4
0051 00000005 0005 00B3 duplicate record 81:5:5
0051 00000005 0006 00B3 duplicate record 81:5:6
40 05 01 0012 00B7 1 byte stored, 5 byte prefix
.
.
.
00 03 03 000B 0293 3 bytes stored, 3 byte prefix
      303000 pfx '.00'
      393133 0298 key '319'
      511E0D 66 029B pointer 81:29:12 ⑤
      0001 029F entry cardinality 1. ⑥
      0000 02A1 leaf cardinality 0.
.
.
.

```

The following list explains those items in Example 12–12 that are specific to ranked sorted indexes:

① **Flag field**

Used by Oracle Rdb to determine if the entry contains a bitmap of duplicate dbkeys and the length of the cardinality fields.

② **Prefix length**

The length of the prefix from the previous entry. Because this is the first entry in the scroll of a node, this length is 0.

③ Separator length

The length of the separator minus the prefix length. In this example, the separator length is 06.

④ Entry length

The length of the variable length section of the entry (001A). (The variable length section of this entry starts at offset 009D and continues to offset 00B7, which is the beginning of the next entry.) Because the fixed length section of an entry is always 5 bytes, add 5 bytes to the entry length to calculate the total length of an entry.

⑤ Dbkey

When an entry contains duplicates but there are not enough duplicates to overflow the leaf node, the dbkey is null (-1:-1:-1), as shown by the first ⑤ callout. If duplicates overflow the leaf node, the dbkey points to the overflow node.

In a unique entry, the dbkey points to a data record (81:29:12), as shown by the second ⑤ callout.

⑥ Entry cardinality

The total number of entries. This includes the duplicate entries stored within the leaf node and any overflow nodes. The first ⑥ callout shows 8 entries, the second shows 1.

⑦ Leaf cardinality

The number of leaf nodes pointed to by this node (0).

⑧ Reference pointer

The starting point of the byte-aligned bitmap compressed (BBC) index segment.

⑨ Bitmap size and number of records

The length of the bitmap that holds the compressed dbkeys (9 bytes) and the number of entries in this node (8).

12.5.3.2 Hashed Index Node Records

Storage areas with mixed page format can contain the rows of two or more tables and associated hashed indexes. To store rows of two or more tables and associated hashed indexes in the same storage area, use the PLACEMENT VIA INDEX option of the SQL CREATE STORAGE MAP or ALTER STORAGE MAP statements and the STORE clause of the CREATE INDEX or ALTER INDEX statements).

Example 12–13 (Cont.) Storage Area Page with Mixed Page Format Containing Hashed Index Node and Data Storage Records

```

001D 0274 line 7: record type 29 ⑤
00 0001 0276 Control information
.... 32 bytes of static data
3F9980004D4750413736313030000111 0279 data '...00167APGM...?'
C834363130304E4D424D09008889A4FF 0289 data '.p...MBMN00164E'
00 0299 padding '.'

2007 029A line 6: duplicate hash node. ③
.... total duplicate node size: 92
FFFF FFFFFFFF FFFF 029E duplicate overflow -1:-1:-1
0045 00000032 0007 02A6 duplicate record 69:50:7
0045 00000032 0005 02AE duplicate record 69:50:5
0045 00000032 0003 02B6 duplicate record 69:50:3

001D 02F6 line 5: record type 29 ⑤
00 0001 02F8 Control information
.... 38 bytes of static data
A49540004D4750413736313030000124 02FB data '$..00167APGM.@.p'
4C45544D00887D88F44780000086DF6A 030B data 'j&...G&}.MTEL'
C03834323030 031B data '00248A'
00 0321 padding '.'

0042 2005 0322 line 4: bucket for hash index 66 ②
.... total hash bucket size: 32
FFFF FFFFFFFF FFFF 0326 bucket overflow -1:-1:-1
00 032E flags 0
00000003 032F duplicate count 3
FFBE 00000032 0006 0333 duplicate node 66:50:6
06 033B key len: 6 bytes
373631303000 033C key: '.00167'

001D 0342 line 3: record type 29 ⑤
00 0001 0344 Control information
.... 38 bytes of static data
1EB140004D4750413736313030000124 0347 data '$..00167APGM.@±.'
5456474D0089A436152FC00000887E52 0357 data 'R~...Ã/.6p..MGVT'
C03736323030 0367 data '00267A'
00 036D padding '.'

003A 2005 036E line 2: bucket for hash index 58 ②
.... total hash bucket size: 32
FFFF FFFFFFFF FFFF 0372 bucket overflow -1:-1:-1
00 037A flags 0
00000001 037B duplicate count 1
003F 00000032 0001 037F pointer 63:50:1
06 0387 key len: 6 bytes
373631303000 0388 key: '.00167'

```

(continued on next page)

Example 12–13 (Cont.) Storage Area Page with Mixed Page Format Containing Hashed Index Node and Data Storage Records

```

001A 038E line 1: record type 26
00 0001 0390 Control information
      .... 72 bytes of static data
69727461706C694B3736313030000110 0393 data '...00167Kilpatri'
3334310B208574656E614A0420836B63 03A3 data 'ck. .Janet. .143'
6C72614D0520A02E745320656E695020 03B3 data ' Pine St.. .Marl'
04C000463635343330484E12208D776F 03C3 data 'ow. .NH03456F.À.'
      08F0310057C79ED7 03D3 data '.ÇW.l.'
      00 03DB padding '.'

2001 03DC line 0: SYSTEM record
02 000E 03DE 14 bytes in 2 sets/dynamic items
0032 06 03E1 6 bytes, storage set type 50
      12 2E 03E4 next 58:50:2
      00 03E6 owner 57:50:0
0039 07 03E7 7 bytes, storage set type 58
0094 5B 03EA next 66:50:4
      00 03ED owner 57:50:0

FFFFFFFF 03EE snap page pointer -1
00000028 03F2 snap pointer TSN 40
      0000 03F6 MBZ '.....'
00000000 03F8 page sequence number 0
00000000 03FC MBZ '....'

```

The amount of space used by hashed index structures on a data page is determined by the number of hashed indexes defined and stored in the storage area, the number of entries in the hash bucket, the number of duplicate records, and the overhead for each record type. See the *Oracle Rdb7 Guide to Database Performance and Tuning* for estimating the size of a hash structure on a data page.

The hashed index structure contains the items shown in Example 12–13 as callouts 1, 2, and 3, followed by the data rows. All are described in the following list:

❶ SYSTEM record, line 0

The system record has as many entries in it as there are hashed indexes defined and stored on a data page within the storage area. The system record entry of a hashed index requires 6 bytes to 10 bytes (compressed) per pointer to each hash bucket plus 4 overhead bytes. The line index entry actually indicates that the system record contains two pointers for a

total of 18 bytes. The system record provides one level of indirect action in the event that a hash bucket cannot be stored on the target page.

② Hash bucket, lines 2 and 4

Each hash bucket contains a fixed header and one hash element for each unique key value mapped to the page. The fixed header includes the following components for a total of 13 bytes:

- Type—2 bytes and frag flags, 2005 is the type
- ID—2 bytes, logical area number (LNO)
- Overflow pointer—8 bytes
- Flags—1 byte, reserved for future use and always zero

Each entry in the hash bucket requires 12 bytes plus the key size plus the key length plus the null bit vector size. If the key length for `EMPLOYEE_ID` is 1 byte and the key size is 5 bytes long plus 1 byte for the null bit vector or 6 bytes, then the total size of the hash entry is 19 bytes. Each hash element or entry in the hash bucket consists of the following:

- A duplicate count—4 bytes
- A dbkey pointer to the value or to the duplicates node—8 bytes
- The key size—1 byte
- The key value and missing value indicator— $n+1$ bytes
 n represents the key value, which varies in size according to the value of the key and the 1 byte is the missing value indicator.

For example, one hash element for the `EMPLOYEES_HASH` hashed index based on the `EMPLOYEE_ID` column is calculated to require $4+8+1+(5+1)$ or 19 bytes. The fixed header is calculated as 13 bytes. Because there is only one hash element in the hash bucket, the hash bucket size is 13 bytes plus 19 bytes or 32 bytes as shown in line index 2 and line entry 2. The hash bucket size for the `JOB_HISTORY_HASH` hashed index, which also has only one hash element that points to a duplicate node, is 32 bytes as shown in line index 4 and line entry 4. Therefore, for this page, both the `EMPLOYEE_HASH` and `JOB_HISTORY_HASH` hash buckets total 64 bytes in size.

③ Duplicate node record, line 6

If duplicates are allowed, each duplicate node record is 92 bytes. Each duplicate node record contains sufficient space to hold pointers to a maximum of 10 duplicate records; for every 10 duplicate records, another duplicate node record is created and pointed to by the previous duplicate

node record. No duplicate records are allowed in the EMPLOYEES_HASH hashed index. There is one duplicate node record in the JOB_HISTORY_HASH hashed index that contains three pointers to the three JOB_HISTORY rows for employee Janet Kilpatrick and requires 92 bytes.

④ The EMPLOYEES data row for Janet Kilpatrick, line 1

There is one EMPLOYEES row stored on this page. It is 80 bytes in length. It has the potential of being 117 bytes long (uncompressed) plus 5 overhead bytes for a total of 122 bytes. (See the *Oracle Rdb7 Guide to Database Design and Definition* to estimate the number of overhead bytes for data rows.) Line entry 1 indicates that the EMPLOYEES row is actually 72 bytes in size (compressed), including row overhead (number of compression bytes, version number, and number of null bytes) plus 5 bytes of data record overhead for a total of 80 bytes as shown in line index 1.

⑤ The JOB_HISTORY data rows for Janet Kilpatrick, lines 3, 5, and 7

There are three JOB_HISTORY rows that belong to Janet Kilpatrick on this page. Each is 34 bytes long with 4 overhead bytes per row for a total of 114 bytes on the page. Line entries 3, 5, and 7 indicate that JOB_HISTORY rows are 38, 38, and 32 bytes in size, respectively (compressed), including row overhead (number of compression bytes, version number, and number of null bytes) plus 5 bytes of data record overhead. The line index entries indicate that the JOB_HISTORY rows are therefore 43, 43, and 37 bytes long for a total of 123 bytes on the page.

In a hashed index scheme, the key value is converted mathematically to a relative page number in the storage area of a particular table.

A **hash bucket** is a data structure that maintains information about the contents of a row (its **search key**), and a list of internal pointers (called database keys or **dbkeys**) to rows that contain the search key. To access a row by using the hashed index, Oracle Rdb first searches the bucket, finds the appropriate dbkeys based on its search key, and fetches the data row. Hash buckets can only occur in an area that has a system record, which is why hashed indexes must be stored in storage areas with mixed page format.

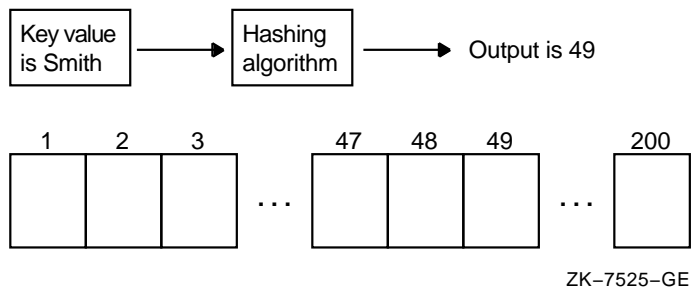
Hashing, also known as hash addressing, provides fast and direct access to a specific row. Access is based on a given value of some set of columns in the row (called the search key). Hashing provides an alternative to the B-tree or sorted index. It is most beneficial for exact-match, direct access when you can supply the entire search key on which the hashed index is defined, such as a social security number. For these types of access, I/O operations can be significantly reduced. This reduction in I/O operations is particularly useful for tables with many rows and large indexes. For example, to retrieve a row by using a sorted index that is four levels deep in the B-tree structure, Oracle Rdb may need to

do five I/O operations. By using hashing, the number of I/O operations may be reduced to one or two.

You can define a search key for both hashed index and sorted index retrieval for the same column. Then, depending on the type of query you use, the Oracle Rdb query optimizer chooses the appropriate method of retrieval. For example, if your query contains an exact-match retrieval, the query optimizer uses hashed index access. If your query contains a range retrieval, the query optimizer uses the sorted index.

Figure 12–2 shows how hashing works, using the PLACEMENT VIA INDEX option for a hashed index where the data and the hashed index are stored in the same physical area. Oracle Rdb takes the search key value and computes a hash value that determines on what target page the row and hash bucket should be placed. In Figure 12–2, page 49 is the hash value Oracle Rdb computes. Because this is the specified target page, this page is checked for space to store the row. The space area management (SPAM) page is not checked first in the case of a hashed index that is used as a placement index to store table rows. If target page 49 has enough free space to store the entire row, Oracle Rdb stores the row on page 49. If there is no available space on target page 49, then each page in the buffer is checked for available space. If there is no available space on any of the pages in the buffer to store the row, then the SPAM page is checked to find a page with available space on which to store the row.

Figure 12–2 Adding a New Row to the Database



Once the row is stored, Oracle Rdb creates an entry in the hashed index. Oracle Rdb hashes the value of the search key again, in case the hashed index is in a different storage area, and returns page 49. Oracle Rdb then fetches page 49 and checks that the system record has an entry for the index in which the first row is stored. If one does not exist, Oracle Rdb creates an entry in the system record. This entry contains the logical area identifier of the hashed

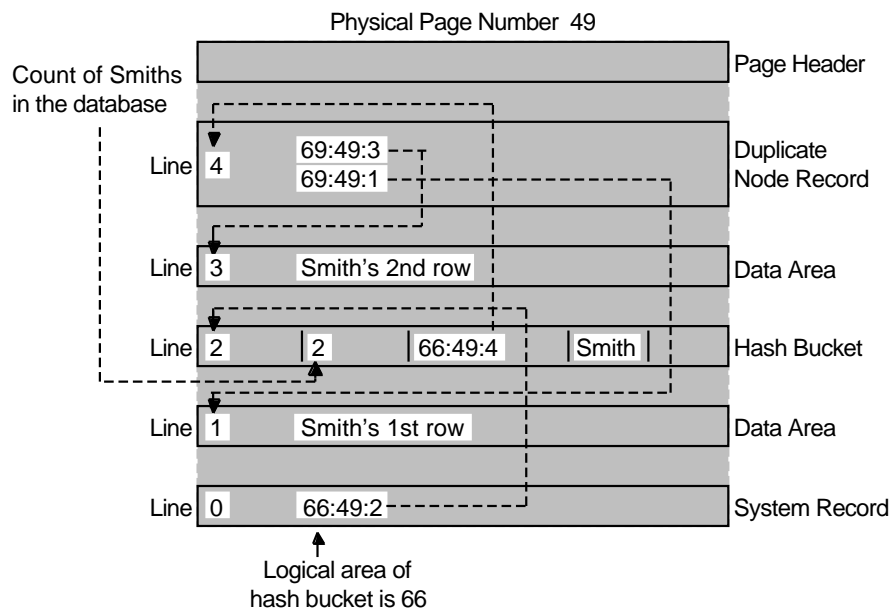
index (in this case, the logical area is 66) and the dbkey of the hash bucket that contains data for the first row. Oracle Rdb then creates a hash bucket that contains the count of the number of data entries of this type, in this case the count of the number of Smiths; the dbkey of the actual data row on page 49; and the search key value of the row just stored.

Therefore, if target page 49 has enough free space to store the hash bucket, Oracle Rdb stores the hash bucket on page 49. If there is no available space on target page 49, each page in the buffer is checked for available space. If there is no available space on any of the pages in the buffer to store the hash bucket, then the SPAM page is checked to find a page with available space on which to store the hash bucket.

Assume you want to store another row that contains the search key value of SMITH. Oracle Rdb again hashes the search key value in the row to determine the target page. If there is enough space on the target page, the row will be stored there. The SPAM threshold value is calculated for the target page. The SPAM page entry for the target page is not checked in order to keep the I/O operations to a minimum. If there is not enough space on the target page, the adjacent pages in the buffer are checked for available space until one is found with sufficient space to store the row. In this instance, if page 49 is the target page and has available space to store another row, then the row is stored there. Because there already is an entry in the system record for the hash bucket that contains the search key value SMITH, when the second Smith row is stored, the pointer to the actual data row is changed to a pointer to a duplicate node record. Oracle Rdb creates a duplicate node record and creates two pointers in the duplicate node record to point to each of the two Smith records. Oracle Rdb changes the pointer in the hash bucket entry to point to this new duplicate node record, and increments the duplicate count in the hash bucket entry.

In Figure 12–3, note that the system record, hashed index, and data area are each contained in separate logical areas on the page. To minimize I/O operations, the SPAM page is not checked initially to determine where there is available space. The SPAM page is only checked as a last resort when the target page and the pages in the buffer are found to have no available space on which to store the row or the hash bucket.

Figure 12–3 Oracle Rdb Page Structure with the System Record



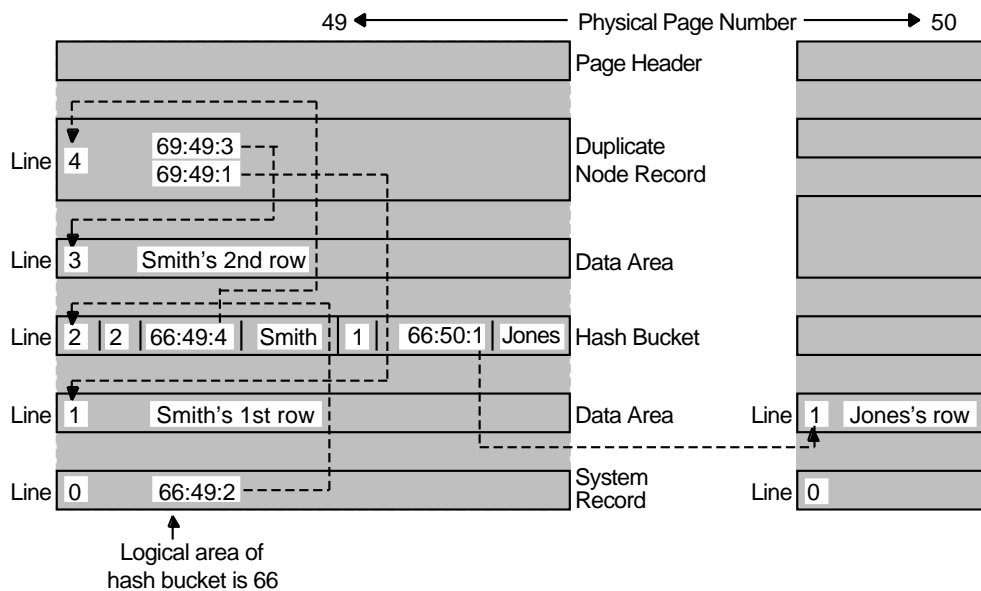
NU-2085A-RA

To retrieve a row, Oracle Rdb does the following:

1. It fetches page 49 and searches the system record for the location of the hash bucket for the correct logical area, in this case, area 66. The system record points Oracle Rdb to the hash bucket for that record type.
2. It then fetches the corresponding hash bucket and finds the correct dbkey among the hash elements or entries stored there along with the pointer to the actual row.
3. It then fetches the row. If the hash element pointer is pointing to a duplicate node, then Oracle Rdb searches the duplicate node for the correct dbkey among the duplicate node entries stored there, finds the one pointing to the actual row, and then fetches the row.

If another value, for example JONES, also hashes to page 49, as in the case of synonyms or aliases, the data row is stored on that page if there is sufficient space, or on a nearby page in the buffer if there is not. In Figure 12–4, the data row is stored on page 50. A new hash element is then created in (or a new entry is added to) the hash bucket on page 49.

Figure 12–4 Pointer to Another Page



NU-2086A-RA

Hashed indexes have the following advantages:

- If you define a hashed index and a table as stored in the same mixed storage area, and the rows in the table are placed using the hashed index, Oracle Rdb stores and retrieves rows in a single I/O operation if the page size and storage area are correctly sized.
- If you assign two different tables to the same storage area by using identical store clauses but different placement indexes in the storage map statements, and both hashed indexes have identical store clauses, Oracle Rdb places rows for both tables and their hashed index entries on the same data page provided that the page size and storage area are correctly sized. Thus, related rows can be read or written to in a single I/O operation.
- If you access rows by using a hashed index, you are less likely to have lock problems because only the index node (hash bucket) that contains the pointer of the row being retrieved is locked. Adjacent key values are unlikely to be stored in the same index node.

12.6 Page Tail for a Data Storage Page

The page tail follows the storage segments at the end of a data page. It has a fixed format that contains 18 bytes on a live page. A live page is a data page and is distinguished from a snapshot (.snp) file page. Example 12–14 shows an example of a page tail in a mixed storage area. Example 12–15 shows an example of a page tail in a uniform storage area.

Example 12–14 Page Tail for a Data Storage Page in a Mixed Storage Area

```

  ❶
0000000E 03EE snap page pointer 14
  ❷
00000016 03F2 snap pointer TSN 22
0000 03F6 MBZ '..'
  ❸
00000000 03F8 page sequence number 0
00000000 03FC MBZ '....'
```

The page tail shown in Example 12–14 contains the following:

- ❶ Snap page pointer
This 4-byte field (0000000E) shows the page number in the .snp file of the most recent .snp file page that contains before-image segments for this data storage page.
- ❷ Snap pointer TSN
This 4-byte field (00000016) shows the most recent transaction that wrote to the snapshot page in item 1.
- ❸ This 4-byte field (00000000) shows the page sequence number.

The remaining fields of the page tail are reserved for Oracle Rdb internal use.

Example 12–15 Page Tail for a Data Storage Page in a Uniform Storage Area

```
  ❶  
000000DA 03EE snap page pointer 218  
  ❷  
00000010 03F2 snap pointer TSN 16  
  ❸  
  0041 03F6 logical area 65  
  ❹  
00000000 03F8 page sequence number 0  
00000000 03FC MBZ '....'
```

The page tail shown in Example 12–15 contains the following:

- ❶ Snap page pointer
This 4-byte field (000000DA) shows the page number in the .snp file of the most recent snapshot page that contains before-image segments for this data storage file page.
- ❷ Snap pointer TSN
This 4-byte field (00000010) shows the most recent transaction that wrote to the snapshot page in item 1.
- ❸ Logical area
This 2-byte field (0041) specifies the logical area for which this page is allocated.
- ❹ Page sequence number
This 4-byte field (00000000) shows the page sequence number. The page sequence number is used for recovery purposes for Oracle Rdb V4.1 and higher.

The remaining fields of the page tail are reserved for Oracle Rdb internal use.

12.7 Fragmented Storage Records

The preceding sections discussed the general format of storage records. A record may be fragmented when stored or modified if there is not enough space on the page to store it. Oracle Rdb fragments the storage record after the storage record header information. This fragmentation is shown in Example 12–16.

Because Oracle Rdb stores a minimum of 10 bytes (2 bytes overhead for the record type plus 8 bytes for the dbkey pointer) for each row, there is always room on the page to convert the storage record or segment into the primary segment of a fragmented record. Oracle Rdb makes certain that the record

is at least the size of a primary segment in case the row is fragmented later. The primary segment contains information (record type and dbkey pointer) that helps Oracle Rdb reassemble a fragmented record. The other segments in the fragmented record, called secondary segments, contain the address (dbkey pointer) of the next segment, if any, and data.

Oracle Rdb may compress the user data portion of the row depending on the characteristics of the row. In any case, the data portion of the segment may look unintelligible when you use the RMU Dump command to look at the on-disk structure. Example 12–16 shows a fragmented record.

Example 12–16 Fragmented Storage Record

```

      ①
0001 00000001 0000 page 1, area 1
      00000000 0006 checksum = 00000000
0094621B 9275A680 000A time stamp = 25-MAR-1995 11:18:31.40
      0000 0000 0012 0 free bytes, 0 locked
      000A 0016 10 lines
      .
      .
      .
      ②
0026 0040 003C line 9: offset 0040, 38 bytes
      .
      .
      .
      ④ ③
      8002 0040 line 9: record type 2
      ⑤
00000003 0006 0042 primary fragment, next is 1:3:6
      ⑥
      00CF 0048 total record length is 207 bytes
      ⑦
010017090A010A011B060002091300CA 004A data '.....'
      29061E060018091B06080108 005A data '.....)'

```

(continued on next page)

Example 12–16 (Cont.) Fragmented Storage Record

```

      .
      .
0001 00000003 0000 page 3, area 1
      00000000 0006 checksum = 00000000
0094621B 9275A680 000A time stamp = 25-MAR-1995 11:18:31.40
      0000 0000 0012 0 free bytes, 0 locked
      0009 0016 9 lines
      .
      .
0046 0074 0030 line 6: offset 0074, 70 bytes
      .
      .
      C002 0074 line 6: record type 2
      0000000A 0002 0076 secondary fragment, next is 1:10:2
3E06001A092C0639062E060019091D06 007C data '.....9,.....>'
001C094E0659064F06001B093C064B06 008C data '.K.<...O.Y.N...''
0A6C0679066D06001D095D0669065E06 009C data '^..i.]...m.y.l.'
      9A0B00210C7C0600890B7D060020 00AC data ' ..}....|!...'
      .
      .
0001 0000000A 0000 page 10, area 1
      00000000 0006 checksum = 00000000
0094621B 9275A680 000A time stamp = 25-MAR-1995 11:18:31.40
      0000 0000 0012 0 free bytes, 0 locked
      000B 0016 11 lines
      .
      .
0028 015A 0020 line 2: offset 015A, 40 bytes
      .
      .
      C002 015A line 2: record type 2
      0000000C 0007 015C secondary fragment, next is 1:12:7
0B00A90B090A00220B008B0B00980B00 0162 data '.....".....'
240C00AC0B00B90B00BB0B00230C009E 0172 data '...#.....$'
      .
      .
0001 0000000C 0000 page 12, area 1
      00000000 0006 checksum = 00000000
0094621B 9275A680 000A time stamp = 25-MAR-1995 11:18:31.40

```

(continued on next page)

Example 12–16 (Cont.) Fragmented Storage Record

```

0000 0000 0012 0 free bytes, 0 locked
          0008 0016 8 lines
          .
          .
002C 0038 0034 line 7: offset 0038, 44 bytes
          .
          .
          C002 0038 line 7: record type 2
0000000F 0007 003A secondary fragment, next is 1:15:7
00DA0B00250C00AE0B00C90B00CA0B00 0040 data '.....*....'
0B00E90B00DE0B00260C00CE0B00D90B 0050 data '.....&.....'
          270C00CF 0060 data '....'
          .
          .
0001 0000000F 0000 page 15, area 1
          00000000 0006 checksum = 00000000
0094621B 9275A680 000A time stamp = 25-MAR-1995 11:18:31.40
          0000 0000 0012 0 free bytes, 0 locked
          0009 0016 9 lines
          .
          .
0034 005E 0034 line 7: offset 005E, 52 bytes
          .
          .
          C002 005E line 7: record type 2
          00000012 0006 0060 secondary fragment, next is 1:18:6
00FF0B00280C00ED0B00F90B00EF0B00 0066 data '.....(.....'
0B01190B010E0B00290C00FD0B01090B 0076 data '.....).....'
          0B01290B011E0B002A0C010D 0086 data '...*.....)..'
          .
          .
0001 00000012 0000 page 18, area 1
          00000000 0006 checksum = 00000000
0094621B 9275A680 000A time stamp = 25-MAR-1995 11:18:31.40
          0000 0032 0012 50 free bytes, 0 locked
          0009 0016 9 lines

```

(continued on next page)

Example 12–16 (Cont.) Fragmented Storage Record

```
      .  
      .  
000B 00A8 0030 line 6: offset 00A8, 11 bytes  
      .  
      .  
      ③  
      4002 00A8 line 6: record type 2  
⑨  
00000001 0009 00AA final fragment, primary is 1:1:9  
01011C 00B0 data '...'
```

The storage segment in Example 12–16 is read from right to left. It contains the following information:

- ❶ Page header
- ❷ Offset from beginning of page to storage segment
The offset, combined with the length entries, identifies the location of a storage segment on the page. The location of a specific storage segment on a page is the sum of the page address and the offset. This location can be represented as:
$$\text{storage-segment-address} = \text{page-address} + \text{offset}$$
- ❸ Storage record ID
Each storage segment is identified by its storage record number. Oracle Rdb uses this internal number to look up the storage segment.
- ❹ Frag flags
The frag flags indicate whether a segment is whole (the value of the flag is 00) or fragmented. If the record has been fragmented, the first bit indicates the presence of a following segment and the second bit, the presence of a preceding fragment. The first segment is called the primary storage segment and the others are called secondary storage segments. In the case of the primary storage segment, the frag flags would indicate 10 (following segment, no preceding segment). A secondary storage segment with a following segment would be 11. The final secondary storage segment would be 01.
- ❺ Fragment chain pointer

The fragment chain pointer consists of the line number and page number where the next segment is located. In the case of all but the last segment in a fragmented row, the line and page number point to the next segment. If the fragment chain pointer is the last in the chain, it points back to the primary segment.

⑥ Total length of storage record

Oracle Rdb uses the length of the storage record to allocate space in virtual memory when it reconstructs a storage record from storage segments in response to a transaction's call for data. Oracle Rdb reassembles the fragments in virtual memory before making the row available to the transaction. It then constructs the row by following the fragment chain pointers to the next segment until it runs out of segments.

The total length of storage record field exists only for the primary segment.

⑦ Uninterpreted data

⑧ Final fragment

Typically, there are several fragmented rows per page. The last page contains the final fragment. In Example 12–16, the final fragment flag is set to 01. In this case, the first bit is a 0 (not displayed) to indicate that this is the final fragment of the fragment chain. The second bit is set to 1 to indicate that there was a preceding segment.

⑨ Uncompressed line number and page number pointing back to the primary segment

Displaying the Contents of SPAM Pages

Oracle Rdb manages free space in a database with space area management (SPAM) pages. A **SPAM page** maintains an inventory of the free space that is available on each data page within an associated physical range of data pages known as the SPAM interval.

By default, there are the following two types of SPAM pages based on the (uniform page or mixed page) format of the storage area:

- SPAM pages for storage areas with uniform page format with preset SPAM intervals and data page fullness thresholds
These include single-file databases and multifile databases with storage areas with uniform page format.
- SPAM pages for storage areas with mixed page format with definable SPAM intervals and data page fullness thresholds
These include multifile databases with storage areas with mixed page format.

The following sections describe the types of SPAM pages in more detail.

13.1 Space Management in Single-File and Multifile Databases

By default, space management is the same for both single-file databases and multifile database storage areas using uniform page format because the storage areas are the same for both. For uniform page format storage areas, Oracle Rdb always knows the size of records that can be stored on a given page because the records are always one size. Consequently, Oracle Rdb maintains simplified SPAM entries for data pages; each SPAM entry indicates whether the data page can or cannot hold one more of the particular record type, that is, there is a full or not full indicator.

Space management in multifile databases for storage areas with mixed page format, however, is different. This type of storage area may contain records of different sizes from more than one table as well as different types of records, such as system records associated with hashed indexes for the tables stored

on the data page. For the mixed page format storage area, space management needs to be much more dynamic, yet offer as much or more efficiency in allocating space between SPAM pages and utilizing space for data on data pages.

To optimize storage efficiency when Oracle Rdb does not know the size of the next record to be stored on the data page, a more efficient mechanism called the fullness threshold value is used to keep track of the percentage of free space left on the data page. This is implemented in one of the following ways:

- By using the THRESHOLDS ARE option that is defined as a storage area option in the SQL CREATE DATABASE statement or the SQL IMPORT statement
- By using the Thresholds qualifier with the RMU Restore, RMU Restore Only_Root, RMU Move_Area, and RMU Copy_Database commands

Note

You must perform a full backup operation immediately after you perform an RMU Move_Area or RMU Copy_Database operation. If parameters are changed during the move or copy operation, the restore and recover operations might not be able to re-create the database correctly.

13.2 Space Management for Logical Areas

Space management for logical areas in storage areas with a uniform page format in both a single-file and multfile database is the same as in a multfile database for storage areas with a mixed page format.

SPAM entries can be one of four possible user-selected values. The default threshold values are (0,0,0), which indicate that the nominal record size should be used for SPAM threshold calculations.

If you use data compression, you should use logical area thresholds. Because uniform page format storage areas in both single-file and multfile databases can contain logical areas with compressed records, this change permits more record storage efficiency with compressed records by allowing you to adjust the space management separately for each logical area in a storage area with uniform page format.

Mixed page format storage areas in multfile databases can contain records of different sizes from more than one table as well as different types of records, such as system records associated with hashed indexes for the tables stored on the data page. For the mixed page format storage area, space management must be dynamic and efficient in allocating space between SPAM pages and utilizing space for data on data pages.

Consider the following methods for optimizing storage efficiency:

- When the size of the next record to be stored on the data page can vary, use the fullness threshold value to keep track of the percentage of free space on the data page.

This threshold value is implemented with the `THRESHOLDS ARE` option that is defined only as a storage map option in the `SQL CREATE` or `ALTER STORAGE MAP` statement and the `SQL IMPORT` statement. You cannot set these threshold values in an `SQL CREATE DATABASE` statement, or `SQL IMPORT` statement, or by using the `Thresholds` qualifier for any `RMU` command except the `RMU Repair` command. See the *Oracle RMU Reference Manual* for more information on using the `RMU Repair` command to set these threshold values.

- Set the interval between SPAM pages.

This is implemented with the `INTERVAL IS` option that is also defined as a storage area option in the `SQL CREATE DATABASE` and `SQL IMPORT` statement. The combination of specifying SPAM threshold values and the SPAM interval for multfile databases with storage areas with mixed page format offers the database administrator (DBA) the needed flexibility and efficiency for allocating space between SPAM pages and utilizing space for data on data pages, both of which have an impact on performance, as described later in this section.

For example, consider SPAM intervals alone. For a 1-block page size, storage areas can have SPAM intervals that range from the default and minimum value of 216 pages to a maximum value of 1964 pages. Compare this to the preset value of 531 pages for single-file databases or multfile databases for storage areas with uniform page format with a default 1-block page size. Definable SPAM intervals help you to estimate the actual space needed for storage areas and reduce SPAM contention.

See the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information on optimizing SPAM intervals.

Combined with other Oracle Rdb features definable SPAM intervals help you tune a database for improved performance by properly defining the page size, the size of the storage area, the page format, the SPAM interval, and the fullness threshold values. These physical design considerations are based on the results of the conceptual and logical database design, the transaction analysis, and the final loaded data model described in the *Oracle Rdb7 Guide to Database Design and Definition*. The Thresholds Are and Interval Is options are described in more detail later in this section.

Section 13.3 and Section 13.4 describe the SPAM pages for both types of storage areas.

13.3 SPAM Pages in Storage Areas with Uniform Page Format

In a single-file database, a SPAM page is the same size as the data storage page it manages. The space between SPAM pages is determined by database page size and clump size. SPAM intervals are as follows:

- Page size 1 block: SPAM interval is 531 database pages
- Page size 2 blocks: SPAM interval is 1089 database pages
- Page size 3 blocks: SPAM interval is 1647 database pages
- Page size 4 blocks: SPAM interval is 2208 database pages

For example, if the page size is 2 blocks, each SPAM page provides information for the next 1089 data pages.

To determine the actual number of pages in the SPAM interval, compare the number of pages addressable by a SPAM page against the maximum number of pages for a uniform page format area. Oracle Rdb uses the smaller of the two values for the number of pages in the SPAM interval.

- To calculate the number of pages that are addressable by a SPAM page:

$$SPAM\ interval\ (database\ pages) = (((S - 24) * 8 - 7) / (P * 2 + 16)) * P$$

In the equation:

S—Specifies the page size in bytes.

P—Specifies the number of pages in a clump.

For example, if the clump size is 12 pages and the page size is 2 blocks or 1024 bytes, then the SPAM interval in pages is as follows:

$$SPAM\ interval\ (pages) = 7993/40 * 12 = 2388\ pages$$

- The following equation calculates the maximum number of pages for a uniform page format area:

$$SPAM\ interval\ (database\ pages) = ((S - 24)/(2 + .25P)) * P$$

For example:

$$SPAM\ interval\ (pages) = 7993/5 * 12 = 19183\ pages$$

Because 2388 is less than 19183, the storage area SPAM interval used is 2388.

Note

To allow for the halfword alignment in the SPAM page for the clump count, you should subtract the value of the clump size (in pages) from the calculated SPAM interval value. This method typically undercounts the actual SPAM interval by as much as the number of pages in a clump and means that sometimes one less clump can be represented on a SPAM page. For example:

Calculated theoretical	clump size	Estimated SPAM
SPAM interval (pages)	- (pages)	= Interval (pages)

$$2388\ pages - 12\ pages/clump = 2376\ pages$$

This calculation is a closer approximation of the estimated SPAM interval and will ensure that you do not undercount the number of SPAM pages in a storage area, especially if the storage area is quite large.

SPAM pages hold a fullness threshold value for each page in a database page interval. By default, each threshold value is a 2-bit entry with a value of 0 or 3; that is, 0 (00) if a page is not full and 3 (11) if a page is full as shown in Example 13-1.

See Table 13-1 for a description and meaning of each of these threshold values.

Example 13–1 (Cont.) SPAM Page for a Storage Area with Uniform Page Format

```

  ③
016B 0127 363 clumps
      ④
4001 0129 pages 2-4, logical area 16385
  ⑤
0001 012B pages 5-7, logical area 1
0002 012D pages 8-10, logical area 2
0003 012F pages 11-13, logical area 3
0004 0131 pages 14-16, logical area 4
0005 0133 pages 17-19, logical area 5
0006 0135 pages 20-22, logical area 6
      .
      .
      .
  ⑥
0000 03FD pages 1088-1090, logical area 0
      ⑦
      00 03FF MBZ free '.'
```

Each SPAM page contains the following information, as shown in Example 13–1:

- ① A standard 22-byte page header (offset 0000–0015).
Unlike a data page, the number of bytes free space is constant and number of bytes of locked free space is zero. Also, all SPAM pages have the most significant bit of the TAD field set to one to indicate this field contains a valid TSN. During an incremental backup operation, the SPAM interval of pages is backed up only if the SPAM TSN is higher than the root file backup TSN providing fast incremental backup performance.
- ② A number n of 2-bit SPAM entries, where n is the SPAM interval and is limited by the capacity of a page.
You designate page size when you create a database with the SQL CREATE DATABASE statement. Oracle Rdb calculates the SPAM interval automatically.
- ③ The count of clumps (016B) mapped on the SPAM page.
Oracle Rdb allocates a group of three database pages into clumps in the same logical area.
- ④ A map (offset 0129–03FF) that indicates to which logical area a page belongs (Logical Area to Page Map).

This map lists:

- ⑤ A logical area entry for a clump of database pages that indicates the logical area to which the clump belongs
- ⑥ A quantity of unused free space shown as logical area 0
- ⑦ SPAM pages have no 10-byte page tail.

SPAM pages are never written to the snapshot (.snp) file when modified.

The order of the 2-bit SPAM entries corresponds to the order of the data pages for which the SPAM page is responsible. For example, the first SPAM entry gives the fullness threshold value for the data page that immediately follows the SPAM page.

Example 13–1 shows the page header, the SPAM threshold values, and the unused portion of the SPAM page. SPAM pages do not count against the storage area's space allocation. They do, however, affect the page numbers of the data storage pages. In Example 13–1, the first data storage page is page 2 and the last data storage page is page 1090. (The page allocation for this area is 1089.)

13.3.1 Area Bit Maps

Storage areas in single-file databases and multifile databases with uniform page format contain a data structure called an area bit map (ABM) that is not contained in storage areas with mixed page format in multifile databases. An ABM page contains **bit vectors** that tell Oracle Rdb which SPAM pages map to which database page clumps for a particular logical area. ABM pages speed up sequential scans of the rows in a table by directing Oracle Rdb to the SPAM page whose bit is set in the logical area's ABM bit vector. That SPAM page is searched to find which clump is allocated to the desired logical area.

Because SPAM pages indicate the fullness threshold of a database page, you can use ABM pages to locate SPAM pages that manage database pages that are not full and can store more rows. ABM pages are also used to speed the gathering of information for each table.

ABM pages are particularly useful for larger databases with more than one SPAM page. In this case, ABM pages prevent Oracle Rdb from reading every SPAM page to find entries for a given logical area.

Example 13–2 displays an ABM page.

13.3.2 Area Inventory Pages

Oracle Rdb uses area inventory pages (AIPs) to maintain a queue of pointers to logical area bit maps (ABMs). Example 13–3 shows the contents of an AIP.

Example 13–3 An Area Inventory Page

```
$ RMU/DUMP/AREA=RDB$SYSTEM/START=2/END=2 MF_PERSONNEL
.
.
.
    ①
0001 00000002 0000 page 2, physical area 1
    EBB7795E 0006 checksum = EBB7795E
009714FB 05FCA7E0 000A time stamp = 20-AUG-1995 15:56:25.31
    0000 0022 0012 34 free bytes, 0 locked
    ②
    00000003 0016 next area inventory page 3
0000000000000000 001A MBZ '.....'
    ③
    0010 0022 16 logical area entries
                                entry #0
    ④
    00000005 0024 first area bitmap page 5
    ⑤
0001 0001 0028 logical area 1, physical area 1
    ⑥
    15 002C area name length 21 bytes
    54535F4445544E454D47455324424452 002D area name 'RDB$SEGMENTED_ST'
    0000000000000000000053474E4952 003D area name 'RINGS.....'
    ⑦
    00000001 004C snaps enabled TSN 1
    ⑧
    00A2 0050 record length 162 bytes
    00000000 0052 MBZ '....'
    ⑨
    01 0056 entry is in use
000000000000000000 0057 MBZ '.....'
    ⑩
    000000 0059 thresholds are (0,0,0)
    000000 005C MBZ '...'
```

(continued on next page)

Example 13–3 (Cont.) An Area Inventory Page

```

                                entry #1
                                00000008 005F first area bitmap page 8
                                0001 0002 0063 logical area 2, physical area 1
                                0D 0067 area name length 13 bytes
000000534E4F4954414C455224424452 0068 area name 'RDB$RELATIONS...'
                                00000000000000000000000000000000 0078 area name '.....'
                                00000001 0087 snaps enabled TSN 1
                                00A8 008B record length 168 bytes
                                00000000 008D MBZ '....'
                                01 0091 entry is in use
                                00000000000000000000000000000000 0092 MBZ '.....'
                                ⑩
                                000000 0094 thresholds are (0,0,0)
                                000000 0097 MBZ '...'
                                .
                                .
                                .

```

The AIP shown in Example 13–3 contains the following:

- ❶ Page header (offset 0000–0015).
- ❷ A pointer (00000003) to the next AIP.
If another AIP does not exist, the value shown is –1.
- ❸ The count (0010) of logical area entries on the AIP.

The following items shown in Example 13–3 are logical area entries:

- ❹ A pointer (00000005) to the first ABM page for a logical area.
- ❺ A physical area identifier (0001) and logical area identifier (0001).
- ❻ The logical area name (54535F4445544E454D47455324424452) and length (15) of the name.
- ❼ The last transaction sequence number (TSN) (00000001) to enable snapshot (.snp) files for a particular logical area or the most recent transaction that did exclusive updates.
- ❽ The buffer length (009A), which corresponds to the record size of the area.
- ❾ A flag (01) that indicates if the entry for the logical area is being used.
If the flag is null, the transaction ID (TID) of the user who deleted the table or index is shown.
- ❿ The SPAM threshold values (000000) for the logical area in the RDB\$SYSTEM storage area with uniform format pages.

Unlike a data page, the number of bytes free space is constant and number of bytes-locked free space is zero. Also, all SPAM pages have the most significant bit of the TAD field set to one to indicate this field contains a valid TSN. During an incremental backup operation, the SPAM interval of pages is backed up only if the SPAM TSN is higher than the root file backup TSN providing fast incremental backup performance.

- ② A number n of 2-bit SPAM entries, where n is the SPAM interval and is limited by the capacity of a page.

You can designate the SPAM interval and page size when you create a database with the SQL CREATE DATABASE statement. Oracle Rdb calculates the SPAM interval automatically.

- ③ SPAM pages have no 10-byte page tail.

SPAM pages are never written to the .snp file when modified.

The order of the 2-bit SPAM entries corresponds to the order of the data pages for which the SPAM page is responsible. For example, the first SPAM entry gives the fullness threshold value for the data page that immediately follows the SPAM page.

Example 13–4 shows the page header, the SPAM threshold values, and the unused portion of the SPAM page. SPAM pages do not count against the area's space allocation. They do, however, affect the page numbers of the data storage pages. In Example 13–4, the first data storage page is page 2 and the last data storage page is page 217. (The page allocation for this area is 216.) A default SPAM interval of 216 pages allows for the creation of a 1-block page size and a 1-block buffer size.

Before data is stored in a mixed page format storage area (.rda) file, Oracle Rdb calculates the threshold of a data page that is guaranteed to hold the record. If a table has a storage map that specifies a PLACEMENT VIA INDEX clause, the specified index is used to determine the target page and Oracle Rdb fetches the target page. Otherwise, Oracle Rdb determines a potential target page for the row. Next, Oracle Rdb checks the threshold of the target page to see if it has sufficient space to fit the record on the page. Oracle Rdb calculates the SPAM threshold of the data page by inspecting its free space and locked space counts in the page header. The SPAM page is not checked for free space in order to save an I/O operation and because the data page is already in the buffer. Finally, the record is *only* stored on the page if the calculated threshold value indicates that there is sufficient space on the page to store the record.

Oracle Rdb does not store a record on a page with a threshold value of 3. In this way, the value you set for the highest threshold can be used to reserve space on the page for future record growth. Free space is guaranteed on the page based on the specified SPAM threshold values. Therefore, the first and second threshold values can be set to store records selectively without having to worry whether or not a PLACEMENT VIA INDEX clause in a storage map takes record storage precedence.

If the potential target data page does not contain enough free space for the entire row, Oracle Rdb continues to search for a data page with sufficient space by scanning only the SPAM pages in the storage area. This scanning provides good performance for locating free space even when the database is nearly full. Without SPAM pages, Oracle Rdb would have to search each data page sequentially until it found a block of free space.

If the potential target data page does not contain enough free space for the entire row, and if a table has a storage map that specifies PLACEMENT VIA INDEX, then the pages in the buffer are first searched for free space. If a page in the buffer has sufficient space to store the record, the record is stored. If no space is found among the pages in the buffer, then Oracle Rdb begins to scan the SPAM pages in the storage area for sufficient space to store the record. Searching the pages in the buffer first, before scanning the SPAM pages, improves performance by reducing I/O operations.

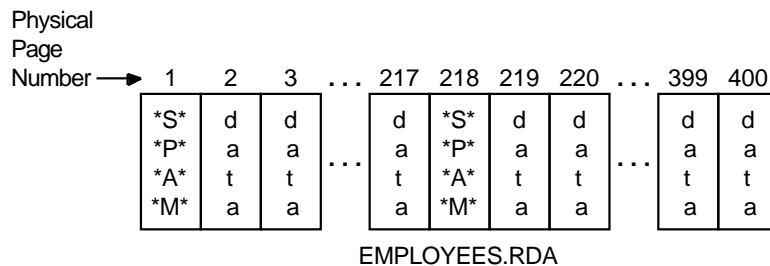
You can control how many data pages each SPAM page manages (the interval of SPAM pages in each storage area file) and you can set three threshold values associated with the SPAM page free space inventory list. By default, these options are available only with multifile database storage areas that use mixed page format. You can specify SPAM threshold values for logical areas in storage areas with uniform format pages in a storage map statement.

In a given .rda file, SPAM pages are located at regular intervals. You control the range of data pages maintained by each SPAM page with the INTERVAL IS option in the SQL CREATE DATABASE statement or in the SQL IMPORT statements. The INTERVAL IS option specifies the number of data pages between SPAM pages in the storage file, and thus the maximum number of data pages each SPAM page will manage.

By default, each SPAM page maintains an inventory of current free space for the next 216 data pages. A SPAM page is the first physical page in a storage area. Following the first SPAM page and its 216 data pages is a second SPAM page. Again by default, the second SPAM page maintains an inventory of free space for the next range of 216 data pages, and so on. SPAM pages do not contain data, and data pages do not contain SPAM information. Figure 13-1

illustrates the physical file layout for a storage area defined with default values (including ALLOCATION IS 400 PAGES).

Figure 13–1 SPAM Intervals in a Mixed Storage Area Using Defaults

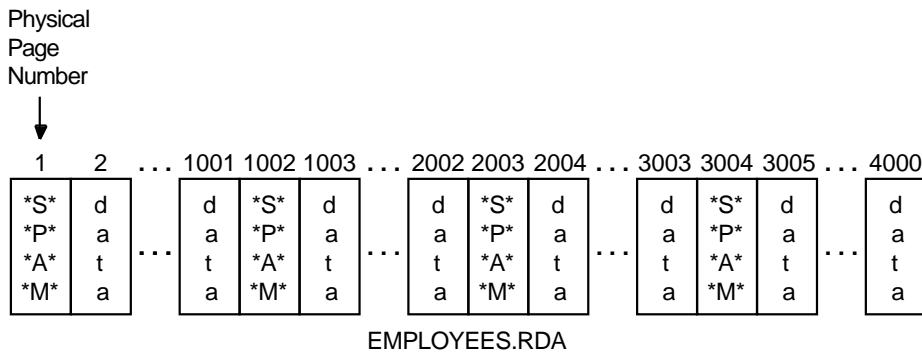


ZK-7526-GE

In Figure 13–1, the first SPAM page (physical page number 1) maintains an inventory of free space that resides on physical data pages 2 through 217. The second SPAM page on page 218 keeps the free space inventory list for pages 219 through 400. Because this EMPLOYEE.RDA file was defined using the default allocation of 400 pages and a SPAM interval of 216 pages, only two SPAM pages were created.

In Figure 13–2, the storage area file allocation is defined as 4000 pages (ALLOCATION IS 4000 PAGES) and the interval of data pages between SPAM pages set to 1000 (INTERVAL IS 1000).

Figure 13–2 SPAM Page Distribution with a Larger Interval



ZK-7527-GE

Four SPAM pages were created in Figure 13–2. Each maintains a free space inventory list as follows:

- The first SPAM page contains the free space inventory list for physical pages 2 through 1001 (1000 pages).
- The second SPAM page contains the free space inventory list for physical pages 1003 through 2002 (1000 pages).
- The third SPAM page contains the free space inventory list for physical pages 2004 through 3003 (1000 pages).
- The fourth SPAM page contains the free space inventory list for physical pages 3005 through 4000 (996 pages). The final range of data pages is limited by the remaining number of pages in the storage area.

The default interval value, 216 pages, is also the minimum value you can specify. The maximum number of entries that fit on a SPAM page (maximum value) depends on the size of the SPAM page. A SPAM page is the same size as a data page. The larger the data page, the larger the SPAM page, and the more entries that fit. The SPAM interval can be no larger than the value calculated from the following formula:

$$((Block\ size * 2048) - 88)$$

Each entry in a SPAM page is 2 bits long (4 entries to the byte), and a SPAM page contains other internal information that occupies 22 bytes. Use the following formula to determine the maximum interval value (the maximum number of data pages for which each SPAM page can hold entries):

$$((Blocks\ per\ page * 512) - 22) * 4$$

The value 512 in the formula represents the number of bytes per block. The value 22 represents the number of bytes in an internal structure called a page header. The value 4 represents the number of SPAM entries per byte.

If your database has 1 block per page, the following formula shows that the maximum number of pages you can specify for the INTERVAL IS option is 1964:

$$((1 * 512) - 22) * 4 = 1964$$

13.5 SPAM Pages in Storage Areas with Mixed Page Format Without a Placement Index

You can control when Oracle Rdb updates a SPAM page's inventory information with the THRESHOLDS ARE option in the SQL CREATE DATABASE statement and in the SQL IMPORT statements. Oracle Rdb keeps track of how much free space is left on a data page by using fullness threshold values. A **fullness threshold** is a percent value that corresponds to the amount of used data page space.

A SPAM page entry for a particular page is updated when the amount of data on that page exceeds a fullness threshold value. You can specify one, two, or three values. If you omit the THRESHOLDS ARE option from the SQL CREATE DATABASE statement, the defaults are 70, 85, and 95 percent. If you specify only one or two values, unspecified values default to 100 percent.

How these SPAM threshold values relate to the storage algorithm is explained in the following sample for the case when no placement index is used to store table rows. For more information on how a hashed index placement index works relative to placing rows on a page, see Section 12.5.3.2.

This particular sample is different from the real sample mf_personnel database described throughout this book. This special sample scenario is used to show how SPAM thresholds work when rows are stored without the aid of a placement index. Assume a user creates a multfile version of the personnel database called mf_personnel. For each table, the user creates a separate .rda file. Default fullness threshold values (70, 85, and 95 percent) are used for all the storage area files except one, the EMPLOYEES.RDA file. For this file, fullness threshold values of 70, 80, and 90 percent are specified. Example 13-5 shows the SQL command procedures for a simple database definition.

Example 13–5 Storage Area SPAM Threshold Parameters for Specific Storage Areas

```
CREATE DATABASE FILENAME "DB_DISK:MF_PERSONNEL" PATHNAME
"CDD$COMPATIBILITY:CORP.ACCT"

! Define database-wide attributes:
    DICTIONARY IS REQUIRED
    SNAPSHOT IS ENABLED DEFERRED
    NUMBER OF RECOVERY BUFFERS IS 100

! Define global storage area attributes:
    ALLOCATION IS 500 PAGES
    PAGE SIZE IS 3 BLOCKS
    THRESHOLDS ARE (70, 85, 95) ! Oracle Rdb default
    INTERVAL IS 256
    PAGE FORMAT IS MIXED

! Override global attributes with local attributes for EMPLOYEES rows:
CREATE STORAGE AREA EMP_STOR_AREA
    FILENAME "USD12:[DBS.DATA]EMPLOYEES.RDA"
    THRESHOLDS ARE (70, 80, 90)
    INTERVAL IS 256
    PAGE SIZE IS 2 BLOCKS
    SNAPSHOT FILENAME "USD13:[DBS.SNAPS]EMPLOYEES.SNP"

! Define separate storage files for remaining tables.
! Use global defaults. For clauses not specified globally,
! Oracle Rdb defaults take effect.
CREATE STORAGE AREA JH_STOR_AREA
    FILENAME "USD14:[DBS.DATA]JOB_HISTORY.RDA"
    SNAPSHOT FILENAME "USD15:[DBS.SNAPS]JOB_HISTORY.SNP"

CREATE STORAGE AREA SH_STOR_AREA
    FILENAME "USD16:[DBS.DATA]SALARY_HISTORY.RDA"
    SNAPSHOT FILENAME "USD17:[DBS.SNAPS]SALARY_HISTORY.SNP"
.
.
.
! End the CREATE DATABASE statement with a semicolon.
.
.
.
! Define storage maps here.
;
```

The SPAM entry for a data page in the EMPLOYEES.RDA file is updated every time the amount of data on that page rises above or falls below any one of the specified percent values. In fact, the values you accept as defaults or specify explicitly with the THRESHOLDS ARE option establish ranges of guaranteed free space on each data page, as shown in Example 13–5.

During the process of storing a row in the EMPLOYEES table, Oracle Rdb calculates the SPAM threshold for the row’s target page and determines if the target data page can guarantee enough free space given the following:

- The size of the row being stored
- The guaranteed free space range in which the page is currently set

The SPAM threshold values of 70, 80, and 90 percent in Example 13–5 establish four possible ranges of guaranteed free space on the data pages. At any given moment during database activity, each data page is identified as falling into only one range.

To show which range a data page is in, Oracle Rdb internally uses a 2-bit value. The binary values and their meanings are summarized in Table 13–1.

Table 13–1 SPAM Entry for a Data Page

Binary Value	Meaning
00	Page fullness currently falls into the <i>first</i> range. There is 0% to less than 70% full and a guarantee that 30% of the total free space is available.
01	Page fullness currently falls into the <i>second</i> range. There is 70% to less than 80% full and a guarantee that 20% of total free space is available.
10	Page fullness currently falls into the <i>third</i> range. There is 80% to less than 90% full and a guarantee that 10% of total free space is available.
11	Page fullness currently falls into the <i>fourth</i> range. There is 90% to 100% full and no guarantee of any free space is available.

If the size of the row being stored is 25 percent of a data page in the EMPLOYEES.RDA file, Oracle Rdb looks only at data pages with a SPAM entry of 00 in the inventory list. The target page is the first candidate that Oracle Rdb checks. If the SPAM entry for the target page is not set to 00, Oracle Rdb checks subsequent SPAM entries on this SPAM page (and therefore in this range of data pages managed by this SPAM page) for a 00 entry. When a data page with sufficient space is located in the SPAM inventory list, or in SPAM index, the row is stored on that page.

Note

To display the current binary values on a SPAM page, use the RMU Dump command. If the EMPLOYEES.RDA file has an interval of 1000 data pages and you determined that physical page 1002 is a SPAM page, (similar to the example in Figure 13–2), enter the following command:

```
$ RMU/DUMP/AREA=EMPLOYEES/START=1002/END=1002 MF_PERSONNEL.RDB
```

Consider the SPAM entry for data page 47 in the EMPLOYEES.RDA file. Before any data is loaded, 100 percent of free space on the page is available. A data page defined with the default of 2 blocks per page (1024 bytes) has approximately 976 free bytes after accounting for the overhead of internal structures. In this example, assume there is a SPAM page on physical page 1 that controls the free space on pages 2 through 257. At this point, the SPAM entry for page 47 contains the binary value of 00.

A program stores many rows, and page 47 is the target page for several of them. As each row is stored, the SPAM page is checked to ensure that enough space exists on page 47 to store the row without fragmenting it. After storing this set of rows, only 35 percent of the free space on this page remains (that is, the page is 65 percent full). The SPAM entry for page 47 still contains the binary value of 00 because the page is within the 0 percent to 70 percent full range.

A subsequent program also stores many EMPLOYEES rows, and several of these rows hash to page 47 (as the target page). (See Section 12.5.3 for information about hashing.) Assume that the size of the first row stored by this program is 14 percent of the entire free space possible on a page in this storage area. The storage algorithm tries to place the row on the target page if the SPAM entry for the target page can guarantee at least 14 percent free space. Because this is the case, the first row is stored. The new row increases the fullness threshold percent of the free space region to 79 percent, leaving 21 percent of free space on page 47. Oracle Rdb updates the SPAM entry for page 47 as the fullness threshold is now in the second (70 percent to 80 percent) range. The binary value for the page 47 entry is changed to 01.

The second row is also 14 percent of the entire free space possible on a page in the EMPLOYEES.RDA file, and it too hashes to page 47. Oracle Rdb proceeds with the following steps:

1. Calculates the SPAM threshold value for target page 47 to be 01.

2. Determines that the SPAM threshold value 01 indicates that the fullness percentage of this page falls into the second range (70 percent to 80 percent full, or a guarantee of 20 percent free space).
3. Sees that the 20 percent guaranteed free space is sufficient to store the row, whose size is just 14 percent of the entire free space.
4. Stores the row on page 47.
5. Updates the SPAM entry for page 47, as the fullness threshold of the page increases from 79 percent to 93 percent (from the second to the fourth range). The binary value for the SPAM entry on page 47 is set to 11.

Assume the third row occupies only 5 percent of the entire free space possible on a page in this storage area, and it hashes to page 47. Oracle Rdb will perform the following steps:

1. Calculates the SPAM threshold value for target page 47 to be 11.
2. Determines that the value 11 indicates that the fullness percentage value of this page falls into the fourth range (90 percent to 100 percent full, or no guarantee of any free space).
3. Immediately rules out page 47 as a possibility even though the page might have enough space to store the entire row. (The reason for this points out the purpose of the third threshold value. When the third value is set to 90, as in this example, Oracle Rdb maintains 10 percent free space on each page for padding, in case an updated row already stored on the page increases in size.)
4. Searches subsequent entries on this SPAM page for a data page that can guarantee at least 5 percent free space.

Assume the adjacent SPAM entry for page 48 contained a binary value of 10. This value indicates to Oracle Rdb that there is at least 10 percent free space, which is sufficient to store the entire third row. In fact, the first entry to have a binary value of 00, 01, or 10 would suffice.

A

Handling Bugcheck Dumps

This appendix describes bugcheck dumps and tells you how to report problems to Oracle. In addition, Section A.3 describes how to track down problems. Several examples describe the most commonly occurring problems.

A.1 Submitting Problem Reports

You might encounter an unexpected and unresolvable software error while using Oracle Rdb. This type of error usually produces a bugcheck dump. If you receive a bugcheck dump, you can report the problem as follows:

- Contact your Oracle supercenter or Oracle representative.
- Have your customer system identifier (CSI) available when you report the problem.
- Collect and include supporting documentation and source material to help Oracle diagnose the problem.

A.2 Troubleshooting Oracle Rdb

Oracle Rdb generates a bugcheck dump file whenever an exception error occurs during database processing. A text file is generated that describes the environment, including the following:

- Process and system parameters
- Copies of the call frame and stack
- Copies of other Oracle Rdb internal data structure information

If the bugcheck dump file indicates that a bug in the Oracle Rdb code exists, contact your Oracle supercenter or representative. If possible, provide a copy of the bugcheck dump and other information to reproduce the problem.

A.2.1 Types of Bugcheck Dumps

There are eight types of Oracle Rdb bugcheck dumps as follows:

- SQL interface bugchecks, written to `SYSS$LOGIN:SQLBUGCHK.DMP`
This bugcheck dump file contains information specific to SQL internal errors.
- Oracle RMU bugchecks, written to `SYSS$LOGIN:RMUBUGCHK.DMP`
This bugcheck dump file contains information specific to RMU that may have resulted from a problem with one of the RMU utilities.
- Oracle Rdb run-time services bugchecks, written to `SYSS$LOGIN:RDSBUGCHK.DMP`
This bugcheck dump file contains process-specific information. Typically, this is the best place to determine what went wrong.
- Database recovery (DBR) bugchecks, written to `SYSS$SYSTEM:RDMDBRBUG.DMP`
This bugcheck dump file contains process-specific information for the detached DBR process. This file often indicates file access problems, as shown in the examples in Section A.3, or system resource limitations, such as insufficient system process control block (PCB) slots.
- AIJ log server (ALS) bugchecks, written to `SYSS$SYSTEM:RDMALSBUG.DMP`
This bugcheck dump file contains process-specific information for the detached ALS process. This file often indicates file access problems, as shown in the examples in Section A.3, or system resource limitations such as insufficient system PCB slots.
- AIJ backup server (ABS) bugchecks, written to `SYSS$SYSTEM:RDMABSBUG.DMP`
This bugcheck dump file contains process-specific information for the detached ABS process. This file often indicates file access problems, as shown in the examples in Section A.3, or system resource limitations, such as insufficient system PCB slots.
- Monitor bugchecks, written to `SYSS$SYSTEM:RDMBUGCHK.DMP`
Monitor bugs (these are very rare) do not produce a bugcheck file; they are written to the monitor log file, which contains process-specific information for the monitor process.
- A monitor log file, written to `SYSS$SYSTEM:RDMMON.LOG`

This is the monitor log file used by the RDMS_MONITOR process. It contains information about database attaches and detaches, recovery processes startup and completion, and other database system log messages.

You can examine the log file after entering an RMU Monitor Reopen_Log command, or use your favorite editor in read-only mode to examine the log file contents while it is being written.

Immediately useful information can be extracted from the bugcheck dump (.dmp) files by using the operating system SEARCH command shown in Example A-1.

Example A-1 Using the SEARCH Command to Find the Exception in an Oracle Rdb Run-Time Services Bugcheck Dump File

```
$ SEARCH /WINDOW SYS$LOGIN:RDSBUGCHK.DMP "*** Exception"
```

A.2.2 Locations of Bugcheck Dump Files

By default, RMU and Oracle Rdb run-time services dump files are written to the directory defined by the SYS\$LOGIN logical name for the process that receives the bugcheck. DBR, ALS, and ABS bugcheck dumps are always written to the directory defined by the SYS\$SYSTEM logical name. When Oracle Rdb produces a bugcheck dump, an error message similar to that shown in Example A-2 is displayed.

Example A-2 Error Message for an Oracle Rdb Run-Time Services Bugcheck Dump

```
%RDMS-F-BUGCHECK, fatal, unexpected error detected  
%RDMS-I-BUGCHKDMP, generating bugcheck dump file SYS$LOGIN:RDSBUGCHK.DMP;1
```

In Example A-2, the bugcheck dump occurred during an Oracle Rdb run-time services operation, as the RDS prefix to the file name (RDSBUGCHK.DMP) indicates.

Example A-3 shows an error message received while running RMU or an application program.

Example A-3 Error Message for an RMU Bugcheck Dump

```
%RDMS-F-BUGCHECK, fatal, unexpected error detected  
%RDMS-I-BUGCHKDMP, generating bugcheck dump file SYS$LOGIN:RMUBUGCHK.DMP;1
```

If a bugcheck dump occurs during a DBR operation, you may receive one bugcheck dump file for each detached recovery process that failed.

A.2.3 Defining the RDM\$BUGCHECK_DIR Logical Name

When you define the RDM\$BUGCHECK_DIR logical name at the system level with SYSNAM privilege, you cause Oracle Rdb to write all RMU and Oracle Rdb run-time services bugcheck dumps to a common directory. DBR, ALS, and ABS bugcheck dumps by default are written to SYSS\$SYSTEM. However, when you define the RDM\$BUGCHECK_DIR logical name, DBR, ALS, and ABS bugcheck dumps will also be written to the new location specified by this logical name.

When users reach their disk quotas in SYS\$LOGIN due to a bugcheck dump, and if the RDM\$BUGCHECK_DIR logical name is not specified to another device, the bugcheck dump overflows to the KOD\$TT file. In this case, a DBR process is created with an output device of KOD\$TT. Also, bugcheck dumps are written to the system disk, and, if the system disk becomes full, the overflow may also end up in KOD\$TT. Under normal circumstances, nothing is written to KOD\$TT and therefore the operating system TYPE command shows that nothing is in the KOD\$TT file.

Define the RDM\$BUGCHECK_DIR logical name so that you can tell by looking in a single directory if a database user received a bugcheck dump. Because bugcheck dump files contain dumps of current database page buffers, these files may contain sensitive data you should protect. Defining the RDM\$BUGCHECK_DIR logical name helps ensure the security of your database and prevents accidental deletions of important bugcheck dump files. Users must have appropriate OpenVMS privileges and quotas to write to the specified output directory and device.

Define the RDM\$BUGCHECK_DIR logical name as shown in Example A-4. Once defined, all RMU and Oracle Rdb run-time services bugcheck dumps are written to the SYS\$MANAGER directory, including DBR, ALS, and ABS bugcheck dumps.

Example A-4 Defining the RDM\$BUGCHECK_DIR Logical Name

```
$ DEFINE/SYSTEM RDM$BUGCHECK_DIR "SYS$MANAGER:"
```

If you use the RDM\$BUGCHECK_DIR logical name to direct output from bugcheck dumps to a common directory, you can set protection on the common directory to make sure no bugcheck dump is ever deleted accidentally before you have a chance to examine it and, if necessary, submit it to Oracle Rdb.

A.3 Understanding Error Messages and Bugcheck Dump Exceptions

Usually, a bugcheck dump indicates that something is wrong with the current processing environment, and not with the database software itself. On occasion, a restriction may be the cause of the bugcheck dump, required files cannot be accessed, the process may have insufficient process quotas, or some system resources may be exhausted. The following sections describe typical problems, the bugcheck dump information that results, and recovery from the problem.

A.3.1 The %RDMS-F-TERMINATE Error

The following error message is issued by the Oracle Rdb monitor whenever a process tries to invoke a database that cannot be recovered:

```
%RDMS-F-TERMINATE, database recovery failed --- access to database denied  
by monitor
```

In some cases, database recovery involves starting a detached process that reads the recovery-unit journal (.ruj) file and performs a rollback on the user's behalf. The process runs the image SYSSYSTEM:RDMDBR, and has a process name that starts with RDB_RB_n.

A database that cannot be recovered is one in which a rollback cannot be performed by the database system. The rollback cannot be performed because the data is in an inconsistent state. Any access should be prevented so users do not read or modify data that is not normally visible.

If the following error message is received by any database user, examine the dump file RDMDBRBUG.DMP in the system directory of the cluster node on which the error was reported:

```
Database recovery failed --- access to database denied by monitor
```

A database cannot be recovered for the following reasons:

- The .ruj file could under certain circumstances remain in the user's directory rather than in the top-level directory on the device in which login information is defined. After a system failure or process deletion, this file has been deleted or renamed, or it is on a disk that has not yet been remounted.

When the database is created (using SQL or Oracle Rally), you can specify the maximum number of users allowed to concurrently access the database. This is specified using the NUMBER OF USERS IS option in the SQL CREATE SCHEMA statement. Each user is allocated space in the database root (.rdb) file to record such information as the .ruj file specification.

You can use the RMU Dump Users command to view this information.

- The after-image journal (.aij) file cannot be extended by the recovery process. This may indicate that no remaining disk quota or no free space remains on the target disk.
- The .aij file cannot be located. This may occur because a systemwide logical name used originally in the after-image journal file name is either not currently defined or has been changed to refer to a different disk or directory.

Example A-5 shows exception reports extracted from RDMDBRBUG.DMP files.

Example A-5 Exception Reports Extracted from RDMDBRBUG.DMP Files

```
**** Exception at 00004360 : DBR$RECOVER + 00000236
%RDMS-F-FILACCERR, error opening recovery-unit journal file
$DUA0:[RDM$RUJ]DB$0097232208D5D240.RUJ;1
-RMS-E-FNF, file not found

**** Exception at 00013616 : UTIO$READ_FILE + 000000F2
%RDMS-F-FILACCERR, error reading disk file
-SYSTEM-F-VOLINV, volume is not software enabled

**** Exception at 00004E4E : AIJ$OPEN + 000001EA
%RDMS-F-FILACCERR, error opening after-image journal file
$DUA1:[CHELSEA]E003.AIJ;1
-RMS-E-FNF, file not found
```

(continued on next page)

Example A-5 (Cont.) Exception Reports Extracted from RDMDBRBUG.DMP Files

```
***** Exception at 00012873 : UTIO$MODIFY_EOF + 00000235
%RDMS-F-FILACCERR, error extending file
-SYSTEM-W-DEVICEFULL, device full - allocation failure
```

A.3.2 Exceeding Quotas

Many problems can be detected and corrected by examining the process-level bugcheck dump file, RDSBUGCHK.DMP. The most common cause of Oracle Rdb bugcheck dumps is the “Exceeded Quota” problem. This section looks at two types of quotas and the reasons why they were exceeded.

A.3.2.1 Disk Quota Exceeded

While processing a query, Oracle Rdb may require temporary files. While using the OpenVMS Sort/Merge utility (SORT/MERGE), Oracle Rdb may require disk space for sorting. The latter case results in the exception shown in Example A-6.

Example A-6 Exception Generated from Exceeding the Disk Quota

```
***** Exception at 001A2B53 : SOR$LIB$SIGNAL + 00000009
%SORT-E-WRITEERR, error writing DISK3:[SHILOH]SORTWORK1.TMP;
-SYSTEM-F-EXDISKQUOTA, disk quota exceeded
```

Oracle Rdb uses SORT/MERGE to perform operations that require an ordered set of records. These operations include the SQL ORDER BY and DISTINCT clauses, and join operations. SORT/MERGE is also used when a sorted index is defined using the SQL CREATE INDEX statement, or during an SQL IMPORT operation.

You can use the logical name RDMSS\$BIND_SORT_WORKFILES to specify how many work files SORT/MERGE is to use if work files are required. The default is 2 work files (the SORT/MERGE default) and the maximum is 10 work files. The location of the work files can be individually controlled by the SORTWORK n logical names (where n is a number from 0 to 9). Refer to the OpenVMS documentation set or online help for more details.

You need to specify these logical names before you run a query if the amount of data being accessed during the query or during index creation does not fit in a work file in your default directory. Some performance improvement can be achieved by distributing the sort work files onto separate, lightly loaded disks.

The batch command procedure shown in Example A-7 includes logical name definitions that enable four work files to be created, each on a separate disk.

Example A-7 Batch Command Procedure to Create Four Work Files, Each on a Separate Disk Volume

```
$ DEFINE RDMS$BIND_SORT_WORKFILES "4"  
$ DEFINE SORTWORK0 DISK1:  
$ DEFINE SORTWORK1 DISK2:  
$ DEFINE SORTWORK2 DISK3:  
$ DEFINE SORTWORK3 DISK4:  
$ RUN END_OF_MONTH_REPORT
```

SORT/MERGE might not use all the work files, so many SORTWORK logical names can be defined as safeguards against unexpected exceptions during database processing.

If the disk quota on the default disk is exceeded, the bugcheck dump is written to the user's terminal. This may indicate that the exception was in fact "disk quota exceeded."

A.3.2.2 Process Quota Exceeded

The following process quotas can also cause exceptions:

- Lock queue limit (ENQLM) too low
- Insufficient page file quota (PGFLQUOTA)
- Insufficient virtual memory (VIRTUALPAGECNT)

Often, as in Example A-8, the error messages themselves do not give many clues to which quota was exceeded. The condition returned to Oracle Rdb by the OpenVMS system contains insufficient information to correctly report all the quotas that are exceeded.

Example A–8 Exception Generated from Exceeding the Paging File Quota (PGFLQUOTA)

```
***** Exception at 002A4DD7 : KOD$GET_VM + 0000026B
%RDMS-F-EXQUOTA, exceeded quota
-SYSTEM-F-EXQUOTA, exceeded quota
```

In these cases, check the name of the routine to diagnose the problem. In Example A–8, the routine `KOD$GET_VM` indicates that the Oracle Rdb executive attempted to expand the process virtual memory (VM) and failed. You should check that the process has sufficient page file quota (PGFLQUOTA). The bugcheck dump file also reveals that this process had 10,000 pages allocated for PGFLQUOTA, which is usually too small for most applications.

A.3.3 Using an Invalid dbkey in an Update Transaction

If you use an invalid database key (dbkey), for example, `dbkey=0`, or if your dbkey scope is `commit` and you delete a record, commit the transaction, and then attempt to use that same dbkey again as a variable in your program in another transaction, Oracle Rdb produces a bugcheck dump rather than return an appropriate error code and message. Oracle Rdb only guarantees that the dbkey points to the same record for the life of the transaction in which the dbkey is retrieved when the dbkey scope is `commit`. Oracle Rdb currently does not support checks for invalid dbkeys.

To remedy this restriction with the `SQL DBKEY SCOPE IS TRANSACTION` option, set the dbkey scope to `ATTACH` when you attach to the database. You can use the same dbkey as a variable by other transactions in your program. When the scope is `ATTACH`, Oracle Rdb guarantees that the dbkey points to the same record until you detach from the database with a `FINISH` statement. For more information on the proper use of the `DBKEY SCOPE IS` option when you attach to the database, see the *Oracle Rdb7 SQL Reference Manual*.

A.4 Reporting a Bugcheck Dump

In general, whenever you receive a bugcheck dump, you should report it to Oracle Rdb. There are some cases, however, when either the bugcheck dump was not caused by a software error, or you may be able to solve the problem first.

A.4.1 Getting a Bugcheck Dump

Bugcheck dumps are caused by the following conditions:

- Setting the dbkey value to an invalid number such as 0
- Setting OpenVMS system parameters improperly

See the *Oracle Rdb7 Release Notes* and the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information on setting these parameters.

- Setting user account parameters improperly

See the *Oracle Rdb7 Release Notes* and *Oracle Rdb7 Guide to Database Performance and Tuning* for more information on setting user account parameters.

- Encountering a software error

A.4.2 Examining a Bugcheck Dump

When you examine the bugcheck dump file, pay careful attention to the name of the database being accessed. You can examine the bugcheck dump file by printing it out or by using an OpenVMS text editor as shown in Example A-9.

Example A-9 Using an OpenVMS Text Editor to Read the Bugcheck Dump File

```
$ EDIT/READ RMUBUGCHK.DMP
```

Search the file for a string of five asterisks (*****). The five asterisks mark the location in the file of the exception code. The exception code describes the error that caused the bugcheck dump. If you see an OpenVMS operating system error such as PAGRDERR or PARITY, your problem is most likely a hardware, rather than a software error (see the OpenVMS documentation or online help for a complete list of OpenVMS operating system error messages). Try to resolve the error by correcting the problem with your hardware.

If the error message indicates that database corruption caused the bugcheck dump, you must restore your database from the most recent full backup copy. See Chapter 7 for more information on restoring your database. See Chapter 5 for information on verifying the integrity of your database once it is restored and on possible causes of database corruption.

You can also examine the system error log to determine why the database was corrupted. Look for memory, bus, controller, and disk errors. Check the system error logs generated since the corrupted page of the database was last modified.

If there is any other error message in your bugcheck dump file, contact your Oracle representative.

A.4.3 Contents of a Bugcheck Dump

An Oracle Rdb bugcheck dump includes a KODA portion with context related to the record storage system (RSS) and a client portion with context related to Oracle Rdb.

The KODA portion contains the following elements:

- A header
- A stack dump
- System information (SYI) and job process information (JPI)
- Client (KODCLIENT) information
- Root file information (in TROOT)
- TROOT global buffer data structures
- Global buffer journal (GBJ) context
- User process block, both KODA and CLIENT
- Stream context
- Object manager (OBJMAN) information (access to root file sections in memory)
- Locking information, both OpenVMS locks and lock cache
- AIJ context and RUJ context
- Scan blocks
- Data I/O (DIO) (segment) context
- Page I/O (PIO) (page) context
 - PIO physical area control block (PACB) context
 - PIO buffer control block (BCB) context
 - I/O control block (buffer pool) (IOCB) context
 - PIO batch-write context
 - Page buffers
- Security audit context
- Virtual memory (VM) utilization

- KODA recovery utility (AIJ rollforward) (KUTREC) context
- AIJ backup context
- System lock information (GETLKI output)

The client (Oracle Rdb) portion contains the following elements:

- A header
- A dump-specified item if RDMS\$DEBUG_FLAGS is defined
- A data definition language (DDL) buffer dump if DDL is active when the dump occurs
- Database (DB) handles of databases attached
- A request block
- List (segmented string) information
- A database definition (SDBD) block
- Symbol tables for tables, indexes, and constraints
- A ready status vector for logical area (SLID)
- A run-time table block (STBL)
- A run-time relation map block (STBL map)
- Run-time version blocks and symbol table per table (SVER)
- A column block per table (SCOL)
- An index block per table (SNDX)
- Index map blocks (SNDX-MAP)
- A run-time collating block (SCOLLATE)
- Query optimization blocks
- Execution structures

This information is useful to an Oracle Rdb software engineer trying to reproduce the error and solve the problem. Especially useful are the KODA header and stack dump portions of the bugcheck dump. The KODA header portion describes the version of Oracle Rdb that was running, what type of machine and the software revision level, the time the bugcheck occurred, and the compiled (build) time of Oracle Rdb components. The stack dump denotes the names of the routines that were executing at the time of the bugcheck dump. The key information is the prefix of the routine name for the routine that was running when the bugcheck dump occurred.

Index

A

- ABM page, 13–8
 - uniform page format, 13–8e
- Abnormal termination
 - of transaction, 10–4
- Access control entry (ACE)
 - for OpenVMS and Oracle RMU, 8–2
- Access control list (ACL)
 - OpenVMS and Oracle RMU ACE, 8–2
- After-image journal (.aij) file
 - allocation and extent size, 9–14
 - avoiding switchover suspension, 9–36
 - backing up
 - multiple fixed-size
 - to disk, 9–26
 - single extensible
 - to tape, 9–25
 - causes for an inaccessible journal file, 9–75
 - checkpointing and backing up, 9–53
 - contents, 9–77
 - disk space requirements, 9–57
 - displaying contents, 9–77
 - fast commit processing and journal file switching, 9–57
 - location, 9–9
 - optimizing, 9–52
 - overview, 9–8, 9–10
 - recovering a lost extensible file, 9–76
 - recovering a lost fixed-size file, 9–77
 - recovery, 9–58, 9–59
 - specifying number of blocks, 9–12
 - switchover, 9–35
 - using disk and tape media, 9–20
- After-image journal file backup
 - using disk and tape media, 9–20
- After-image journaling
 - AIJ request blocks, 9–7
 - avoiding switchover suspension, 9–36
 - causes for an inaccessible journal file, 9–75
 - devising a strategy, 9–4
 - disabling, 9–8
 - enabling, 9–5
 - extensible journals, 9–10
 - fixed-size journal file switchover, 9–35
 - journal file, 9–8, 9–10
 - modifying during a restore operation, 8–40
 - multiple fixed-size journal files, 9–10
 - procedures, 9–1
 - recommended and required usage, 9–2
 - recovering a lost extensible journal file, 9–76
 - recovering a lost fixed-size journal file, 9–77
 - strategy for list data on WORM media, 9–16
 - transaction sequence number, 9–80
 - usage with the Commit To Journal option, 9–58
 - usage with the fast commit option, 9–53
 - using the RMU Dump After_Journal command, 9–77
- AIJ request blocks
 - require extra OpenVMS pages, 9–7
- AIP
 - uniform page format, 13–10e
- Alarms
 - Audit type, interpreting, 3–18
 - Daccess type, interpreting, 3–18
 - Oracle RMU type, interpreting, 3–20
 - Protection type, interpreting, 3–20
 - security, 3–3, 3–17

- ALLOCATION IS option, 9–12
- ALTER DATABASE statement, 10–6
 - JOURNAL EXTENT IS option, 9–5
 - NUMBER OF RECOVERY BUFFERS, 10–6
- Altering corrupt databases, 6–3
- Area bit map
 - See* ABM page
- Area inventory page
 - See* AIP
- ASTLM parameter
 - values, 8–36
- Attach command (RdbALTER), 6–4
- Attaching to a database, 4–5
 - in RdbALTER, 6–4
- Audit
 - See* Audit events
- Audit events
 - disabling Daccess, 3–5, 3–9
 - disabling event information, 3–12
 - enabling Daccess, 3–5, 3–9
 - enabling event information, 3–12
- Auditing
 - audit event information
 - enabling or disabling, 3–12
 - Daccess level security auditing, 3–8, 3–9
 - event-level security auditing, 3–8, 3–12
 - four levels of security auditing, 3–7
 - monitoring resources, 3–4
 - specific objects, 3–9
 - specific object types, 3–9
 - starting, 3–13
 - stopping, 3–8, 3–13
 - top-level security auditing, 3–8, 3–13
 - user-level security auditing, 3–7, 3–9
- Audit journal records, 3–3, 3–22
 - defining database table for storing, 3–22
 - defining relation for storing, 3–22
 - reviewing, 3–24
- Audit trail
 - in RdbALTER, 6–25
- Availability features, 1–8
 - cluster environment automatic recovery, 1–10
 - disable logging for write-once storage areas, 1–10

- Availability features (cont'd)
 - fault-tolerant Rdb databases, 1–9
 - maintaining optimum database integrity, 1–15
 - maintenance
 - analyze operations, 1–16
 - automatic cleanup activities, 1–10, 1–15
 - online backup operations, 1–9
 - online DBA activities, 1–11
 - online move operations, 1–10
 - online recovery operations, 1–10
 - online restore operations, 1–10
 - verify operations, 1–16

B

- Backing up a database, 7–1
 - controlling tape concurrency, 7–68
 - optimizing tape utilization, 7–71
 - preloading and synchronizing backup tapes, 7–69
- Backup
 - by-area incremental backup, 7–35
 - by area operation, 7–8
 - by storage area, 7–19
 - complete incremental backup, 7–35
 - database backup (.rbf) file, 7–9
 - Digital UNIX tar, 7–13
 - displaying root file information, 7–77
 - examples, 9–82
 - read-only and updatable storage areas, 7–42
 - sample procedure, 7–26
 - file management, 7–19
 - frequency, 7–16
 - full, 7–8, 7–32e
 - full and complete, 7–17
 - full versus incremental, 7–18
 - incremental, 7–8, 7–38
 - comparing timestamps, 7–38
 - disk I/O costs, 7–41
 - performance, 7–37
 - Scan_Optimization qualifier, 7–37
 - selecting pages, 7–38
 - summary statistics, 7–41
 - to tape, 7–36

Backup (cont'd)

- incremental sample commands, 7-35
- introduction to Oracle RMU backup, 7-3
- multiple parallel processes, 7-4
- multithreaded, 7-2
- off line, 7-22
- on line, 7-56
- on line versus off line, 7-22
- OpenVMS Backup utility, 7-13
- options, 7-8
- parallel, 7-23
- performance
 - calculate working set, 7-29
- quota-exceeded problems, 8-36
- read/write storage areas, 7-22
- read-only and write-once storage areas, 7-20
- read-only storage areas, 7-21
- single CPU process, 7-3
- specifying file protection, 7-13
- strategies, 7-15, 9-19
- to disk media, 7-61
- to tape, 7-34
- to tape media, 7-62
- trade-offs
 - incremental backup versus multiple journal files, 9-19
 - traditional backups, 7-2
 - underrun errors, 7-71
 - using shadowed or mirrored disks, 7-15
 - when to back up journal files, 9-22

Backup After_Journal command

- tape label checking, 7-73

Backup command

- See* RMU Backup command

Before-image journal

- See* Recovery-unit journal (.ruj) file

Bit vectors, 13-8

Bugcheck dump

- (ABS) RDMABSBUG.DMP file, A-2
- (ALS) RDMALSBUG.DMP file, A-2
- causes, A-7, A-10
- contents of, A-11
- controlling placement of, A-4
- database recovery (DBR) process, A-3
- (DBR) RDMDBRBUG.DMP file, A-2

Bugcheck dump (cont'd)

DBR process

- output device KOD\$TT, A-4

error messages, A-5 to A-9

- disk quota exceeded, A-7
- insufficient page file quota (PGFLQUOTA), A-8
- insufficient virtual memory (VIRTUALPAGECNT), A-8
- invalid dbkey, A-9
- lock queue limit (ENQLM) too low, A-8

examining, A-10

example error messages, A-6

exception code, A-10

locations, A-3

Oracle Rdb run-time services, A-3

Oracle RMU, A-3

reporting, A-9

(RMU) RMUBUGCHK.DMP file, A-2

(Run-time services) RDSBUGCHK.DMP file, A-2

(SQL) SQLBUGCHK.DMP file, A-2

SYSSYSTEM:RDMBUGCHK.DMP file, A-2

types of, A-2

typical message, A-3

By-area backup

- guidelines for file management and recovery, 7-19

incremental, 7-44

read-only and write-once storage areas, 7-44

strategies, 7-19

strategies for read/write storage areas, 7-22

strategies for read-only and write-once storage areas, 7-20

strategies for read-only storage areas, 7-21

timestamps, 7-41

updatable storage areas, 7-44

C

CDD

information

- moving during restore, 8-49

- Changing database page contents, 6–12
- Changing the radix in RdbALTER, 6–13, 6–25
- Checking for database corruption, 5–19
- Checking tape labels with Oracle RMU, 7–73
- Checksum, 7–73, 12–3
- Checksum_Verification qualifier, 7–31
- Clearing an inconsistent flag, 6–23
- Closing a database, 4–1, 4–7
 - for a maintenance operation, 4–9
- Cluster
 - after-image journal file placement, 9–9
 - closing a database, 4–14
 - cluster-accessible journal files, 10–3
 - opening a database, 4–3
- Commit command (RdbALTER), 6–15, 6–26
- Committing a transaction
 - in RdbALTER, 6–26
- Commit To Journal option
 - usage with after-image journaling, 9–58
- Common data dictionary
 - information
 - duplicating during restore, 8–48
 - during restore, 8–51
- Completing a transaction
 - in RdbALTER, 6–15, 6–26
- Constraints
 - checking, 5–4, 5–7
 - checking with RMU Verify, 5–9
 - verifying, 9–85
 - violating a definition, 5–3
- Contents
 - ABM page, 13–9
 - AIP, 13–10, 13–11
 - data page with hashed index, 12–26
 - display and interpret
 - file contents, 11–1, 13–1
 - storage pages, 12–1
 - hashed index node record, 12–28
 - index node record, 12–21
 - journal file, 9–77
 - line index, 12–4
 - lists, 12–10
 - locked and unlocked free space, 12–6
 - page header, 12–3
 - page tail

Contents

- page tail (cont'd)
 - mixed storage area, 12–35
 - uniform storage area, 12–36
- .ruj file, 10–8
- snapshot page, 11–18
- SPAM page, 13–7, 13–12
- storage segment, 12–8
- TSN, 12–5
- user-stored records, 12–9
- Coordinator processes, 7–5
- Copying a database into a directory owned by a
 - resource identifier, 8–2
- Corruption
 - causes of database corruption, 5–2
 - checking for database corruption, 5–3
 - clearing a corruption flag, 6–5
 - clearing an inconsistent flag, 6–23
 - corrupt page table, 5–21, 5–32, 9–60
 - detecting, 5–19
 - example, 5–33
 - data integrity, 5–43
 - page header, 5–33
 - sorted index, 5–38
 - in batch-update mode, 5–3
 - patching the database, 6–3
 - repairing the database, 6–1
- Corrupt page table, 5–21, 5–32, 8–19, 9–60
- CREATE DATABASE statement, 10–6, 13–12
 - NUMBER OF RECOVERY BUFFERS IS
 - option, 10–6
 - SPAM pages, 13–12
- Creating databases
 - space management, 13–12
 - tailoring the system, 13–12
- Cyclic redundancy check (CRC)
 - See also* Underrun errors
 - automatic error correction, 7–72
 - avoiding underrun errors, 7–71
 - Checksum option, 7–73
 - end-to-end error detection, 7–73
 - options, 7–72

D

Daccess audit events

enabling or disabling, 3-5, 3-9

Data

moving, 6-17

storage page header, 12-3

storage page structure, 12-1

Database

attaching to, 4-5, 4-6

backing up

by-area operation, 7-8

checking tape labeling, 7-73

controlling tape concurrency, 7-68

efficient use of multiple tape drives, 7-67

example, 7-26, 7-32, 7-34

frequency of, 7-16

full backup operation, 7-8

incremental, 7-35

incremental operation, 7-8

incremental storage area backup, 7-44

multiple tape drives, 7-65

multithreaded backup operation, 7-2

off line, 7-22

on line, 7-56

optimizing tape utilization, 7-71

preloading and synchronizing backup

tapes, 7-69

quota-exceeded problems, 8-36

read-only storage areas, 7-44

recommendations for, 7-25

sample procedure, 7-26

single tape drive, 7-63

to tape, 7-34

types of, 7-8

updatable storage areas, 7-44

write-once storage areas, 7-44

backups, 7-1

causes of corruption, 5-2

changing page contents, 6-12

characteristics, 2-16

checking for corruption, 5-3

checkpointing and backing up journal files,

9-53

Database (cont'd)

Checksum_Verification qualifier, 7-31

closing, 4-1, 4-7, 4-11

for maintenance operations, 4-9

corruption causes, 5-2

corruption example, 5-33

data integrity, 5-43

page header, 5-33

sorted index, 5-38

creating

See Creating databases

database administrator (DBA), 1-1

database recovery (DBR) process, 10-4

design space management, 13-12

determining when it is open, 4-3

displaying and interpreting

file contents, 11-1, 13-1

page contents, 6-10

security auditing characteristics, 3-2

storage pages, 12-1

displaying header information, 7-77

duplicate database, 8-47

example of default verify operation, 5-17

example of full verify operation, 5-17

improving verify performance, 5-16

integrity, 5-4

journal file performance, 9-52

journaling, 10-1

management requirements, 1-2

monitoring, 2-1

moving files, 6-16, 8-49

opening, 4-1, 4-2, 4-6

by attaching, 4-6

optimizing journal file, 9-52

options file, 8-45

page content changing, 6-12

pages

restore example, 8-18

preparing to restore, 8-1

read-only and updatable storage areas

backup example, 7-42

considerations, 7-42

restore example, 8-14

recovery, 7-1, 8-1, 9-58, 10-1

reorganizing single to multifile, 8-52

Database (cont'd)

- reporting bugcheck dumps, A-9
- restoring, 8-1
 - checking database version numbers, 8-13
 - creating duplicate database, 8-47
 - disabling SPAM pages, 8-44
 - enabling SPAM pages, 8-44
 - example, 8-13
 - full restore operation, 8-5
 - incremental restore operation, 8-10
 - modifying after-image journaling, 8-40
 - modifying database characteristics, 8-37
 - modifying options, 8-45
 - modifying page size, 8-43
 - modifying thresholds, 8-43
 - moving dictionary information, 8-50
 - moving files, 8-49
 - multiple tape drives, 8-33
 - quota-exceeded problems, 8-36
 - read/write storage areas, 8-18
 - setting snapshot allocation size, 8-44
 - setting the Noworm attribute, 8-44
 - setting the Worm attribute, 8-44
 - single tape drive, 8-32
- restoring into directory owned by a resource identifier, 8-2
- restricting access, 4-3
- root file
 - See Root file
- search key, 12-30
- setting global buffers, 4-3
- snapshot pages, 11-25
- space management, 13-12
- storage page structure, 12-1, 12-2
- strategy to detect problems, 5-9
- table for storing security audit journal records, 3-22
- tracking open and close operations, 4-4
- user information, 2-12
- verification troubleshooting, 5-21
 - checksum check, 5-23
 - data integrity corruption, 5-25
 - summary, 5-31
- verifying, 5-1
 - problems detected, 5-7

Database

- verifying (cont'd)
 - reasons for, 5-1
 - what happens, 5-3
 - what is checked, 5-4, 5-12
 - verifying constraints, 5-48
- Database administrator (DBA)
 - primary tasks, 1-1, 1-2
- Database automatic recovery process
 - improving performance, 10-6
- Database backup (.rbf) file, 7-9
 - keeping incremental backup operations, 7-28
 - purging, 7-28
- Database corruption
 - after system failure, 5-20
 - causes, 5-2
 - checking for, 5-3
 - clearing a corruption flag, 6-5
 - clearing an inconsistent flag, 6-23
 - detecting, 5-19
 - example, 5-33
 - data integrity, 5-43
 - page header, 5-33
 - sorted index, 5-38
 - in batch-update mode, 5-3
 - patching, 6-3
 - repairing, 6-1
- Database design
 - space management, 13-12
- Database files
 - moving, 6-16
 - storage pages, 12-1
 - types of files, 13-1
- Database key (dbkey)
 - components of, 12-4
 - compressed, 12-23
 - hashed indexes, 12-26, 12-30
 - overflow, 12-33
 - index node segments, 12-20
 - sorted indexes
 - duplicate index node, 12-23
 - uncompressed, 12-23
- Database page
 - ABM pages, 13-8
 - AIPs

- Database page
 - AIPs (cont'd)
 - uniform page format, 13–10e
 - area bit maps
 - uniform page format, 13–8e
 - checksum, 12–3
 - common format used in RMU Dump output, 11–2
 - displaying contents, 6–10
 - fragmented record, 12–36, 12–37
 - free space, 12–3, 12–6
 - fullness threshold, 13–1
 - hashed index, 12–26
 - structure with page pointer, 12–33
 - header, 12–3
 - index node record, 12–18, 12–20, 12–23
 - line index, 12–4
 - list record, 12–10
 - locked and unlocked free space, 12–4, 12–6, 12–7
 - mixed page format
 - SPAM intervals, 13–15f
 - SPAM page, 13–12e
 - number, 12–3
 - page tail
 - mixed storage area, 12–35
 - uniform storage area, 12–35
 - snapshots, 11–25
 - space area management (SPAM) pages, 13–5e
 - SPAM entry information, 13–19t
 - SPAM interval formula, 13–16
 - storage area number, 12–3
 - storage segment structure, 12–8
 - structure with system record, 12–32
 - timestamp, 12–3
 - TSN index, 12–5
 - units, 12–1
 - user-stored record, 12–9
- Database record
 - hashed index, 12–28
- Database recovery (DBR) process, 10–4
 - creating, 10–5
- Database reorganizing
 - single to multiframe, 8–52
- Database storage page structure, 12–1
- Database verification, 5–4
 - after changes in RdbALTER, 6–25
 - after system failure, 5–2
 - after using RdbALTER, 6–15
 - database page checking, 5–19
 - effects on other users, 5–4
 - example of default verify operation, 5–17
 - example of full verify operation, 5–17
 - excluding all data, 5–9
 - frequency to perform, 5–4, 5–9
 - full, 5–4, 5–9
 - how it works, 5–25
 - improving verify performance, 5–16
 - including all data, 5–9
 - incremental, 5–17
 - optional before backup and after restore operations, 5–1
 - problems detected, 5–7
 - problem suspected, 5–2
 - purpose, 5–1
 - resource constraints, 5–2
 - restrictions on READY mode, 5–4
 - specific areas, 5–38
 - strategy, 5–9
 - troubleshooting, 5–21
 - checksum check, 5–23
 - data integrity, 5–25
 - summary, 5–31
 - what happens, 5–3
 - what verify checks, 5–4, 5–12
- Data compression
 - storage efficiency with logical area thresholds, 13–2
- Date and time stamp
 - See* Timestamp
- DBA
 - See* Database administrator (DBA)
- Dbkey
 - See* Database key (dbkey)
- DBR process
 - See* Database recovery (DBR) process

- DECnet network protocol
 - restriction when restoring a database, 8-1
- Deposit command (RdbALTER), 6-12
- Detach command (RdbALTER), 6-5, 6-27
- Detaching from a database
 - in RdbALTER, 6-5
- Digital UNIX
 - LSM mirrored disks, 7-15
 - tar, 7-13
- DIOLM parameter
 - values, 8-36
- Directories
 - recovery-unit journals, 10-2
- Disabling audit event information, 3-12
- Disabling Daccess audit events, 3-5, 3-9
- Disk
 - incremental backup command, 7-35
- Disk backups
 - guidelines, 7-61
 - recommendations for after-image journal file backup, 9-20
- Disk device
 - backing up after-images, 9-20
 - reusing after-image backup media, 9-21
- Disk space requirements
 - fast commit transaction processing enabled and after-image journal backup, 9-57
 - journal file size, 9-88
- Display command (RdbALTER), 6-10
- Displaying database page contents, 6-10
- Displaying user information, 2-12
- Dump After_Journal command
 - tape label checking, 7-73
- Dump Backup command
 - tape label checking, 7-73
- Duplicate index nodes, 12-19, 12-23
- Duplicate node record
 - hashed index structure, 12-28

E

- Enabling audit event information, 3-12
- Enabling Daccess audit events, 3-5, 3-9

- Error
 - See* Bugcheck dump
- Errors
 - monitoring tape drives, 7-76
 - submitting software problem reports, A-1
 - underrun, 7-71
- Exception condition
 - See* Bugcheck dump
- Exclude qualifier
 - omitting specific storage areas, 7-44
- Exit command (RdbALTER), 6-27
- EXPORT statement
 - when to use, 8-52
- Extensible journals, 9-10

F

- Fast commit option
 - usage with after-image journaling, 9-53
- Fast commit processing
 - effect on journal file switchover, 9-57
- Fast commit transaction processing enabled and after-image journal backup
 - disk space requirements, 9-57
- Fault tolerance, 1-9
- Fetching a page (RdbALTER), 6-7
- Fetching a storage area (RdbALTER), 6-6
- Fixed-size journal files
 - effect of fast transaction processing, 9-57
- Fragmentation
 - frag flags, 12-9
 - record, 12-36 to 12-41
 - storage records, 12-36
- Fragment chain pointer, 12-41
- Free space
 - database page, 12-7
 - find with SPAM pages, 13-12
 - locked, 12-4, 12-6
 - unlocked, 12-6
- Full database backup, 7-32
 - comparing with incremental backup, 7-18, 7-41
 - requirements for using, 7-17
 - starting, 7-32

Full database verification, 5–9
Fullness threshold, 13–1, 13–17

G

Global buffers
 showing information on, 2–12

H

Hash bucket record
 hashed index structure, 12–28
Hashed index
 duplicate node record, 12–28
 hash bucket record, 12–28
 node records, 12–25
 SYSTEM record, 12–28
Help command (RdbALTER), 6–27
Help facility, xxiii
Help on line, xxiii

I

IMPORT statement, 10–6
 NUMBER OF RECOVERY BUFFERS IS
 option, 10–6
 when to use, 8–52
Inconsistent flag
 clearing, 6–23
Incremental database backup
 comparing with full backup, 7–18, 7–41
 determining which pages have changed, 7–38
 for selected storage areas, 7–44
 measuring benefits, 7–41
 optimizing performance, 7–37
 sample command lines, 7–35
 starting, 7–34
Incremental verification, 5–17
Index
 duplicate nodes, 12–19, 12–23
 nodes, 12–19
 overflow nodes, 12–19
Index node record, 12–18
 data region length, 12–22
 example, 12–20, 12–23

Index node record (cont'd)
 fragmentation, 12–20
 index scroll, 12–22
 level type, 12–22
 storage record type ID, 12–22
 uncompressed owner dbkey, 12–22
Index node segment, 12–8

J

JOURNAL EXTENT IS option, 9–5
Journaling, 10–1
 See also Recovery-unit journal (.ruj) file
 after-image, 9–1
 after-image journal directory, 9–9
 AIJ end-of-file, 9–14
 backing up
 after-images to disk, 9–20
 after-images to tape, 9–20
 a single extensible file, 9–25
 multiple fixed-size after-image to disk,
 9–26
 multiple fixed-size files, 9–26
 reusing after-image backup media, 9–21
 reusing backup media, 9–21
 single extensible after-image to tape,
 9–25
 common directories, 10–2
 devising a strategy, 9–4
 directory protection, 9–9
 disabling, 9–8
 enabling, 9–5
 example, 9–82
 file protection, 9–88
 file size, 9–88
 incomplete recovery, 10–1
 information not written to journal file, 9–3
 information written to journal file, 9–3
 logical end-of-file, 9–14
 logical names, 9–9
 multiple fixed-size files, 9–10
 multiuser access, 10–2
 optimizing after-image journaling, 9–52
 performance, 9–5, 9–9, 9–52, 10–2
 physical end-of-file, 9–14

Journaling (cont'd)

- placement, 9-9
 - recovery, 9-58
 - recovery-unit journal, 10-2
 - relationship between allocation and extent size, 9-14
 - requirements for, 9-1
 - reviewing security audit records, 3-24
 - .ruj file placement, 10-2
 - security audit, 3-3, 3-22
 - setting allocation size, 9-12
 - setting extent size, 9-6
 - single extensible journal file, 9-6, 9-10, 9-14
 - strategy for list data on WORM media, 9-16
 - trade-offs
 - multiple journal files or incremental backup, 9-19
 - truncating journal files, 9-88
- Journal qualifier
- Oracle RMU backup, 7-71

L

- Label
 - checking on tapes, 7-73
- Leaf node, 12-19
- Line index, 12-4
- Lists, 12-8, 12-10
- Live page, 11-22, 11-25
- Loader synchronization
 - preloading tapes for parallel backup operation, 7-54
- Loader_Synchronization qualifier
 - Oracle RMU backup, 7-69
- Loading data
 - security audit journal records, 3-2, 3-22
- Lock
 - duplicate nodes, 12-20
- Locked and unlocked free space, 12-4
- Locking duplicate nodes, 12-23
- Locking quiet point, 7-56
- Log command (RdbALTER), 6-25
- Log file
 - incremental backup summary statistics, 7-41

Logical area thresholds

- storage efficiency with data compression, 13-2

Logical name

- after-image journal file, 9-9
- defining RDM\$BUGCHECK_DIR, A-4
- defining RDM\$BIND_SORT_WORKFILES, A-8
- journal files, 9-9
- recovery-unit journal, 10-2
- system, 9-9

M

Maintenance

- activities requiring reload, 1-15
 - analyze operations, 1-16
 - automatic cleanup activities, 1-10
 - availability features, 1-8
 - closing the database, 4-10
 - cluster environment automatic recovery, 1-10
 - disable logging for write-once storage areas, 1-10
 - general database activities, 1-7t
 - general SQL activities, 1-8t
 - multithreaded backup operation, 7-2
 - offline DBA activities, 1-14
 - online backup operations, 1-9
 - online DBA activities, 1-11
 - online move operations, 1-10
 - online recovery operations, 1-10
 - online restore operations, 1-10
 - overview, 1-3
 - regular activities, 1-4t
 - startup and shutdown activities, 1-5t
 - troubleshooting activities, 1-6t
 - verify operations, 1-16
- Make Consistent command (RdbALTER), 6-23
- Management functions (Oracle RMU)
- privileges required to use, 3-6t
- Master qualifier
- Oracle RMU backup, 7-68
- mf_personnel database
- creating sample, 1-16

- Migrating databases
 - to multfile database, 8–52
- Monitoring a database, 2–1
- Monitor log file
 - RDMS_MONITOR process information, A–3
 - reopening, 2–7
 - See also* Monitor process
 - log file
- Monitor process, 2–2
 - attaching to a database, 2–8
 - changing base priority, 2–5
 - changing log file, 2–5
 - changing priority, 2–7
 - communicating with users, 2–8
 - displaying, 2–6
 - displaying contents, 2–6
 - log file
 - activity information, 2–10
 - creating new version, 2–5
 - displaying, 2–5, 2–6, 4–4
 - example, 2–8
 - header information, 2–11
 - monitor bugs, A–3
 - reading, 2–8
 - renaming, 2–5
 - SYSSYSTEM:RDMMON.LOG file, A–3
 - priority, 2–6
 - recording user activity, 2–8
 - recovery processes, 2–7
 - starting, 2–2, 2–3
 - stopping, 2–2, 2–3
 - with active users, 2–5
 - using mailboxes, 2–8
- Move command (RdbALTER), 6–17
 - to move database files, 6–17
- Moving database files, 6–16
- Multiple fixed-size journal files, 9–10
- Multithreaded backup
 - introduction, 7–3
 - parallel backup, 7–4
 - strategies, 7–23

N

- New features
 - Oracle RMU, xxvii
- New features for Oracle Rdb, xxvii
- Nodes
 - index, 12–19
- Nolog command (RdbALTER), 6–26
- NUMBER OF RECOVERY BUFFERS IS option
 - determining the current value, 10–7
 - specifying with SQL, 10–6

O

- Object types
 - auditing, 3–9
 - auditing for specific privileges, 3–9
- Offline backup
 - strategies, 7–22
 - when to perform, 7–22
- Online backup
 - enabling snapshot files, 7–56
 - performing, 7–56
 - starting, 7–58
 - strategies, 7–22
 - when to perform, 7–22
- Online help, xxiii
- Opening a database, 4–1
- OpenVMS
 - shadowed disks, 7–15
- OpenVMS Backup utility, 7–13
- OpenVMS security audit journal, 3–16
 - loading into database, 3–22
- Optimizing after-image journaling, 9–73
- Oracle Rdb Management Utility
 - See individual RMU command entries*
- Oracle Rdb new features, xxvii
- Oracle RMU
 - privileges applied when you restore a database, 8–2
 - privileges required to use, 3–6t
- Oracle supercenter
 - submitting problem reports, A–1

Overflow index nodes, 12-19

P

Pages

- checksum, 12-3
- corrupt page table, 5-21, 5-32, 9-60
- data structure, 12-1
- header, 12-3
- number, 12-3
- SPAM

- See* Space area management (SPAM) pages

- tail, 12-35

Parallel backup

- assigning worker processes to nodes, 7-47
- coordinator process, 7-5
- overview, 7-4
- Parallel Backup Monitor, 7-47, 7-55
- performing, 7-45
- plan file, 7-49
- starting, 7-48
- steps during processing, 7-52
- strategies, 7-23
- using with loader synchronization, 7-54
- worker processes, 7-5

Parameters

- ASTLM, 8-36
- DIOLM, 8-36

Patching database corruption, 6-3

Performance

- improving database automatic recovery process, 10-6
- journaling, 10-2
- NUMBER OF RECOVERY BUFFERS IS option, 10-6
- optimizing after-image journaling, 9-52
- optimizing the incremental backup operation, 7-37

PERSONNEL database

- creating sample, 1-16

Primary segment, 12-8

Privilege

- granting after an RMU Restore operation, 8-2

Privilege (cont'd)

- required for accessing recovery-unit journal file, 10-4

Process failure, 10-6

Process quotas

- for Oracle RMU backup, 7-29

Protection audit events

- enabling or disabling, 3-6

Q

Quiet point, 7-56

Quiet-point lock, 7-56 to 7-61

R

Radix command (RdbALTER), 6-13, 6-25

.rbf file

- See* Database backup (.rbf) file

RdbALTER, 6-3

- Area command, 6-6
- Attach command, 6-4
- attaching to a database, 6-4
- changing page contents, 6-12
- Commit command, 6-15, 6-26
- completing transactions, 6-26
- Deposit command, 6-12, 6-14e, 6-15e
- Detach command, 6-5, 6-27
- detaching from a database, 6-5
- Display command, 6-10
 - Area_Number, 6-10
 - asterisk (*), 6-10
 - Checksum, 6-10
 - Count, 6-11
 - Data, 6-11
 - Free_Space, 6-11
 - Header, 6-10
 - Index, 6-11
 - Line, 6-11
 - Locked_Free_Space, 6-11
 - Page_Number, 6-10
 - Space, 6-11
 - Time_Stamp, 6-10

- displaying page contents, 6-10

- entering, 6-4

RdbALTER (cont'd)

- Exit command, 6-27
- exiting, 6-27
- fetching an area, 6-6
- fetching a page, 6-7
- Help command, 6-27
- information needed before using, 6-3
- keeping an audit trail, 6-25
- keeping a session log, 6-4
- Log command, 6-4, 6-25
- logging a session, 6-25
- Make Consistent command, 6-23
- Move command, 6-17
 - example, 6-17
- moving database files, 6-16
- moving data on a page, 6-17
- Nolog command, 6-26
- PAGE command, 6-7
- Radix command, 6-13, 6-25
- Rollback command, 6-26
- specifying an area, 6-6
- specifying a page, 6-6
- stopping a session log, 6-26
- Uncorrupt command, 6-5
- undoing changes, 6-4
- Verify command, 6-25
 - verifying alterations, 6-25
 - with RMU Verify, 6-3
- RdbALTER Verify versus RMU Verify, 6-25
- RDM\$BUGCHECK_DIR logical name defining, A-4
- RDM\$RUJ directory, 10-2
- RDMMON.LOG file, 2-5, 2-6
 - See also* Monitor process log file
- Record
 - clusters, 12-9
 - fragmentation, 12-36 to 12-41
 - growth, 12-8
 - index node, 12-18
 - lists, 12-10
 - user-stored, 12-9
- Recover command
 - tape label checking, 7-73

Recovering a database

- using after-image journal files, 9-58
- Recovery, 10-1
 - See also* Database
 - abnormally terminated transaction, 10-4
 - abnormal termination, 10-4
 - after-image journal files, 9-58
 - allocating number of DBR database buffers, 10-6
 - displaying the value of NUMBER OF RECOVERY BUFFERS, 10-7
 - from a lost extensible journal file, 9-76
 - from a lost fixed-size journal file, 9-77
 - from an inaccessible journal file, 9-75
 - incomplete, 10-1
 - order to apply, 9-68
 - placement of recovery-unit journal files, 10-2
 - recovery-unit journals, 10-1
 - steps for, 9-59
 - strategies, 9-19
 - when to back up journal files, 9-22
- Recovery-unit journal (.ruj) file, 10-4, 10-6
 - directories, 10-2
 - displaying contents, 10-8
 - file
 - creating and protecting, 10-4
 - interpreting contents, 10-8
 - interpreting file headings, 10-8
 - logical names, 10-2
 - missing recovery-unit journal files, 10-1
- Recovery-unit journals
 - creating on common-access devices, 10-3
 - .ruj file location, 10-2
- Relation
 - for storing security audit journal records, 3-22
- Reorganizing databases
 - database files, 8-52
- Repairing database corruption, 6-1
- Reporting
 - notifying Oracle about problems with Oracle Rdb software, A-1
- Restore command
 - tape label checking, 7-73

- Restore Only_Root command
 - tape label checking, 7-73
- Restoring
 - databases, 7-1
- Restoring a database
 - with after-image journaling, 9-2
- Restoring databases, 8-1
 - checking database version numbers, 8-13
 - creating duplicate database, 8-47
 - creating new database version, 8-9
 - DECnet restriction, 8-1
 - disabling SPAM pages, 8-44
 - enabling SPAM pages, 8-44
 - examples
 - no read-only storage areas, 8-13
 - pages, 8-18
 - read-only storage areas, 8-14
 - from tape, 8-32
 - full, 8-5, 8-6
 - granting directory ACE privileges, 8-2
 - granting Oracle RMU privileges, 8-2
 - incremental, 8-10
 - applying restore to correct root file version, 8-12
 - modifying after-image journaling, 8-40
 - modifying database characteristics, 8-37
 - modifying database options, 8-45
 - modifying page size, 8-43
 - moving files, 8-49
 - Noworm attribute, 8-44
 - options file, 8-45
 - preparation, 8-1
 - quota-exceeded problems, 8-36
 - root file, 8-22
 - setting snapshot allocation size, 8-44
 - threshold values, 8-43
 - Worm attribute, 8-44
- RMU Alter command
 - entering RdbALTER, 6-4
- RMU Backup After_Journal command
 - tape label checking, 7-73
- RMU Backup command
 - advantages of, 7-13
 - After_Journal, 9-20
 - using disk and tape media, 9-20

- RMU Backup command (cont'd)
 - by-area backup requires file management, 7-19
 - calculating working set, 7-29
 - Checksum_Verification qualifier, 7-30
 - compared to EXPORT, 8-52
 - Crc=Autodin_ii, 7-72
 - Crc=Checksum, 7-73
 - Crc qualifier requirements, 7-72
 - cyclic redundancy check (crc) qualifier options, 7-72
 - example, 7-34
 - Execute qualifier, 7-46
 - full and complete, 7-17
 - full backup operation, 7-8
 - full versus incremental, 7-18
 - Incremental, 7-34e
 - Incremental=By_Area, 7-35
 - Incremental=Complete, 7-35
 - Journal qualifier, 7-71
 - List_Plan qualifier, 7-46, 7-51
 - Load_Synchronization qualifier, 7-69
 - Lock_Timeout qualifier, 7-60
 - Master qualifier, 7-68
 - multithreaded backup operation, 7-2
 - Noexecute qualifier, 7-49
 - Online qualifier, 7-58
 - on line versus off line, 7-22
 - parallel, 7-23
 - Parallel qualifier, 7-46
 - Protection qualifier, 7-13
 - qualifiers that select storage areas, 7-19
 - Quiet_Point lock qualifier, 7-59
 - Quiet_Point qualifier, 7-56
 - read/write storage areas, 7-22
 - read-only and write-once storage areas, 7-20
 - read-only storage areas, 7-21
 - Scan_Optimization qualifier, 7-37
 - tape label checking, 7-73
- RMU Backup Plan command, 7-46, 7-51
- RMU Close command, 4-7, 4-10e
 - Noabort qualifier, 4-13
 - terminating active users, 4-10

- RMU Dump After_Journal command, 9-77
 - tape label checking, 7-73
- RMU Dump Backup_File command, 7-77
 - tape label checking, 7-73
- RMU Dump command, 11-1
 - After_Journal, 9-77
 - Area qualifier, 11-1
 - display database characteristics, 2-16
 - Header, 2-16
 - Larea qualifier, 11-1
 - Recovery_Journal, 10-8
 - Snapshots qualifier, 11-1
 - to show storage area, page, and record numbers, 6-7
 - users, 2-13e
- RMU Dump Header command, 7-56
- RMU Load Audit command, 3-2, 3-23, 3-24
- RMU Monitor command, 2-2
 - Start, 2-3
 - start monitor and capture the output, 2-6
 - start specifying a priority, 2-6
 - start the monitor and display output, 2-6
 - Stop, 2-3, 2-6
 - stop and abort, 2-3
- RMU Open command, 4-2e, 4-3, 4-7
 - Access qualifier, 4-3
 - Global_Buffer qualifier, 4-3
 - implicit, 9-88
 - Path qualifier, 4-3, 4-9
- RMU Optimize After_Journal command, 9-73
- RMU Recover command, 9-66
 - tape label checking, 7-73
- RMU Repair command, 6-1
- RMU Restore command, 8-6
 - After_Journal, 8-40, 8-41
 - Aij_Options, 8-40, 8-41
 - Blocks_Per_Page, 8-43
 - compared to IMPORT, 8-52
 - DECnet restriction, 8-1
 - Directory, 8-48, 8-49
 - Duplicate, 8-47
 - File, 8-48, 8-49
 - Incremental, 8-10
 - Root, 8-10
- RMU Restore command (cont'd)
 - into a directory owned by a resource identifier, 8-2
 - New_Version, 8-9
 - Noafter_Journal, 8-40, 8-41
 - Noaij_Options, 8-40
 - Nocdd_Integrate, 8-51
 - Nospams, 8-44
 - Noworm, 8-44
 - Only_Root, 8-22
 - Options, 8-45
 - Path, 8-50
 - preparation, 8-1
 - purpose, 8-1
 - Snapshot=(Allocation=N), 8-44
 - Snapshots, 8-48, 8-49
 - Spams, 8-44
 - tape label checking, 7-73
 - Thresholds, 8-43
 - Worm, 8-44
- RMU Restore Only_Root command
 - tape label checking, 7-73
- RMU Set After_Journal command
 - Add, 9-76
 - Disable, 9-8, 9-76
 - Drop, 9-57
 - Enable, 9-5, 9-82
 - Notify, 9-77, 9-82
 - Reserve, 9-82
 - Shutdown_Timeout, 9-40
 - Switch_Journal, 9-24, 9-30, 9-71, 9-82
- RMU Set Audit command, 3-2, 3-9
 - Enable=Daccess=Column qualifier, 3-12
 - Enable=Daccess=Database qualifier, 3-11
 - Enable=Daccess=Table qualifier, 3-11
 - Enable=Daccess qualifier, 3-12
 - Enable=Protection qualifier, 3-12
 - Enable=RMU qualifier, 3-12
 - Every qualifier, 3-14
 - First qualifier, 3-14
 - Flush qualifier, 3-14
 - NoFlush qualifier, 3-14
 - Privileges qualifier, 3-11
 - Start qualifier, 3-15
 - Type=Alarm qualifier, 3-15

- RMU Set Audit command (cont'd)
 - Type=Audit qualifier, 3-15
- RMU Set command
 - Corrupt_Pages Consistent, 6-5, 6-23
- RMU Set Corrupt_Pages command
 - set consistent flag, 6-21
- RMU Set Corrupt_Pages Consistent command
 - clearing an inconsistent flag, 6-23
 - replaces RdbALTER Uncorrupt command, 6-5
- RMU Show Audit command, 3-2
 - Audit qualifier, 3-12
 - Daccess=Column qualifier, 3-12
 - Daccess=Database qualifier, 3-11
 - Daccess=Table qualifier, 3-11
 - Every qualifier, 3-14
 - Flush qualifier, 3-14
 - Identifiers qualifier, 3-9
- RMU Show command
 - Corrupt_Pages, 5-21, 5-32, 9-60
 - system, 2-11
 - users, 2-11, 2-12
 - Users qualifier, 4-3e
 - version, 2-11
- RMU Show Corrupt_Pages command, 6-22
- RMU Show Statistics command, 1-5
 - after-image journal backup files, 9-49
 - after-image journal file switcover, 9-42
 - after-image journaling performance, 9-5
 - after-image journaling realtime information, 9-44
 - after-image journal sequence numbers and checkpoint numbers, 9-48
- AIJ Information and Stall Messages screens, 9-41
- checking for disk bottlenecks during verify operations, 5-17
- emergency after-image journal files, 9-39
- submitting as a batch job, 2-14
- RMU Verify command, 5-4, 5-12
 - All, 5-4, 5-13, 5-14
 - Areas, 5-14, 5-38
 - Checksum_only, 5-13, 5-25
 - Constraints, 5-15
 - Incremental, 5-13

- RMU Verify command (cont'd)
 - Indexes, 5-15
 - Indexes [No]Data, 5-15
 - Larea, 5-14
 - Larea Segmented_Strings, 5-15
 - Log, 5-20
 - qualifier functions, 5-12
 - Root and Noroot, 5-14
 - Snapshots, 5-14
 - Transaction_Type, 5-20
- RMU Verify versus RdbALTER Verify, 6-25
- Rollback command (RdbALTER), 6-26
- Rolling back a transaction
 - in RdbALTER, 6-26
- Root file
 - backup information, 7-38, 7-77
 - corrupt page table, 5-21, 5-32, 9-60
 - header information, 2-16
 - mapping global sections, 4-1
 - mapping shared memory partitions, 4-1
 - restore information, 8-10, 8-12
 - unmapping global sections, 4-1
 - unmapping shared memory partitions, 4-1
- .ruj file
 - See Recovery-unit journal (.ruj) file

S

- Sample database, creating, 1-16
- Scan_Optimization qualifier
 - optimizes incremental backup performance, 7-37
- Search key, 12-30
- Secondary segment, 12-8
- Security
 - database recovery, 9-58
- Security auditing, 3-2
 - alarms, 3-3, 3-17
 - enabling, 3-8
- Audit event type, 3-4
- audit journal, 3-3, 3-22
 - enabling, 3-8
- Daccess
 - event type, 3-5
 - level security auditing, 3-8, 3-9

- Security auditing
 - Daccess (cont'd)
 - privileges for database objects, 3-10
 - default characteristics, 3-2
 - defining audit events, 3-7
 - displaying characteristics, 3-2
 - enabling and disabling events, 3-12
 - establishing auditing, 3-8
 - event-level security auditing, 3-8, 3-12
 - event types, 3-4
 - every access, 3-13
 - first access only, 3-13
 - Flush qualifier, 3-14
 - interpreting alarms, 3-17
 - interpreting Audit alarms, 3-18
 - interpreting Daccess alarms, 3-18
 - interpreting Oracle RMU alarms, 3-20
 - interpreting Protection alarms, 3-20
 - levels of security auditing, 3-7
 - monitoring resources, 3-4
 - NoFlush qualifier, 3-14
 - Protection event type, 3-6
 - reviewing Audit alarms, 3-18
 - reviewing audit information, 3-16
 - reviewing audit journal, 3-24
 - reviewing audit journal records, 3-24
 - reviewing Daccess alarms, 3-18
 - reviewing Oracle RMU alarms, 3-20
 - reviewing Protection alarms, 3-20
 - RMU event type, 3-6
 - setting alarms, 3-13
 - setting Daccess events, 3-9
 - setting record auditing, 3-13
 - setting user-level events, 3-9
 - starting and stopping, 3-8, 3-13
 - strategy for defining security auditing, 3-7
 - top-level security auditing, 3-8, 3-13
 - use of the RMU Load Audit command, 3-22
 - user-level security auditing, 3-7, 3-9
 - Security audit journal records
 - defining database table for storing, 3-22
 - defining relation for storing, 3-22
 - loading into database, 3-22
 - Show System command (RMU), 2-7, 2-12
 - Show Users command (RMU), 2-11
 - Show Versions command (RMU), 2-11
 - Snap page, 11-25
 - Snapshot (.snp) file
 - data structures, 11-25
 - display, 11-18
 - Snapshot files
 - enabling for online backup operations, 7-56
 - Sorted index, 12-20
 - non-ranked, 12-19, 12-20
 - index node segment, 12-20
 - ranked, 12-19, 12-23
 - index node segment, 12-23
 - Space area management (SPAM) pages
 - controlling thresholds, 13-17
 - defaults, 13-12
 - defining, 13-12
 - defining intervals, 13-17
 - entries, 13-1
 - entry information, 13-19t
 - examined during incremental backup, 7-38
 - format, 13-7, 13-12
 - fullness threshold, 13-17
 - mixed page format, 13-12e
 - page interval, 13-1
 - sample scenario, 13-17e
 - SPAM interval
 - formula for uniform storage area, 13-5
 - structure, 13-1
 - threshold value, 13-5
 - uniform page format, 13-4, 13-5e
 - SPAM page
 - See* Space area management (SPAM) pages
 - SQL EXPORT statement, 8-52
 - SQL IMPORT statement, 8-52
 - Starting security auditing, 3-8, 3-13
 - Stopping security auditing, 3-8, 3-13
 - Storage area
 - backup strategies, 7-19
 - mixed page format, 11-6
 - requirements of journal files, 9-88
 - specifying for back up operations, 7-44
 - structures, 12-1
 - uniform page format, 11-3

- Storage records
 - frag flags, 12-9
 - identification, 12-9
 - record pointer, 12-9
 - structure, 12-9, 12-10
- Storage segment
 - fragmented, 12-8, 12-37
 - length, 12-5
 - storage record header, 12-8
 - structure, 12-8
- Summary statistics
 - incremental backup operation, 7-41
- Switching journal files, 9-35
 - avoiding switchover suspension, 9-36
- Switchover, 9-35
- SYSSYSTEM logical name, 2-5
- SYSHUTDOWN.COM command procedure, 2-2
- SYSTARTUP.COM command procedure, 2-2, 2-6
- System failure, 10-6
 - checking database for corruption, 5-20
 - verifying database integrity, 5-2
- SYSTEM record
 - hashed index structure, 12-28

T

- Tape
 - checking labels, 7-73
 - incremental backup command, 7-36
 - monitor error rates, 7-76
 - optimizing utilization, 7-71
 - preloading, 7-69
- Tape backups
 - guidelines, 7-62
 - recommendations for after-image journal file backup, 9-20
 - underrun errors, 7-71
- Tape device
 - backing up after-images, 9-20
 - reusing for after-image backup, 9-21
- Tape label checking, 7-73
- Thresholds
 - fullness, 13-1

- Timestamp
 - by-area backup operation, 7-41
 - changes, 12-3
 - full database restore operation, 8-5
 - incremental backup operation, 7-38
 - incremental restore operation, 8-10
 - page header, 12-3
- Transaction
 - abnormal termination, 10-4
 - journaling, 9-1
 - locked free space, 12-4
 - normal completion, 10-4
 - recovering, 9-66
 - update, 10-4
 - abnormal termination, 10-4
 - exception using invalid dbkey, A-9
- Transaction identification number (TID), 12-6
- Transaction sequence number (TSN), 12-5
 - after-image journaling, 9-80
 - index, 12-5
 - used for incremental backup, 7-38
- TSN
 - See* Transaction sequence number (TSN)

U

- Uncorrupt command (RdbALTER), 6-5
- Underrun errors
 - avoiding, 7-71
- Update transaction, 10-4
 - abnormal termination, 10-4
- User information, displaying, 2-12
- Users
 - listing active, 2-12
 - terminating active, 4-7, 4-10
 - verifying active, 4-13
- User-stored data storage segments, 12-8
- User-stored record, 12-9
- Using the RMU Dump Area command, 11-8, 11-9
- Using the RMU Dump Larea=RDBSAIP command, 11-15
- Using the RMU Dump Larea command, 11-13

Using the RMU Dump Snapshot command,
11-18
compare with live page, 11-22e

V

Verification

databases prior to backup using the
Checksum_Verification qualifier, 7-31

Verify command

checking constraints, 5-9
constraints, 9-85

Verify command (RdbALTER), 6-25

Verifying

checking constraint definitions, 5-4, 5-7
constraints on a restored database, 9-85
violating a constraint definition, 5-3

Verifying database integrity, 5-4, 5-25

after alterations in RdbALTER, 6-25
after system failure, 5-20
after using RdbALTER, 6-15

database page checking, 5-19
effects on other users, 5-4
frequency to perform, 5-4
full, 5-4
restrictions on ready mode, 5-4
specific areas of the database, 5-38

W

Worker processes, 7-5

Working set

calculating for Oracle RMU backup, 7-29

WORM optical media

journaling considerations, 9-16
omitting both backup and journaling
operations, 9-17
performing backups but omitting journaling
operations, 9-17
performing journaling but omitting backup
operations, 9-18

