

Oracle® Rdb7 for OpenVMS

Release Notes

Release 7.0.6

November 2000

ORACLE®

Oracle Rdb7 Release Notes, Release 7.0.6 for OpenVMS

Copyright © 1984, 2000, Oracle Corporation. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs 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. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, 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 52.227-19, Commercial Computer Software - Restricted Rights (June, 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, backup, redundancy, and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle is a registered trademark, and Oracle Rdb, Rdb7, Oracle SQL/Services, Oracle7, Oracle Expert, and Oracle Rally are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

This document was prepared using VAX DOCUMENT Version 2.1.

Contents

Preface	xviii
1 Installing Oracle Rdb7 Release 7.0.6	
1.1 Requirements	1-1
1.2 Invoking VMSINSTAL	1-1
1.3 Stopping the Installation	1-2
1.4 After Installing Oracle Rdb7	1-2
1.5 Alpha Wildfire Processor Support Added	1-2
1.6 Maximum OpenVMS Version Check Added	1-3
2 Software Errors Fixed in Oracle Rdb7 Release 7.0.6	
2.1 Software Errors Fixed That Apply to All Interfaces	2-1
2.1.1 False Reports of RDB-F-OBSOLETE_METADATA	2-1
2.1.2 Bugcheck at RDMS\$\$COMPILE_INDEX_MAPS + 00000447	2-2
2.1.3 Memory Leak in Lock Structures	2-2
2.1.4 Query With MIN Function on a Partitioned Table Returns Wrong Results	2-3
2.1.5 Monitor Bugchecks at MON\$DELETE_UNREFERENCED_GBL	2-3
2.1.6 Recovery Operation Work File Allocation	2-4
2.1.7 Query With Shared Expression in OR Predicate Returns Wrong Results	2-4
2.1.8 Left Outer Join Query With UNIONed Sub-queries Returns Wrong Results	2-5
2.1.9 SPAM Thresholds Incorrectly Set	2-7
2.1.10 ORDER BY Query With IN Clause Returns Wrong Order	2-7
2.1.11 RDMS-F-READ_ONLY Error When Storage Area Mode Changed by Other Node	2-8
2.1.12 Join Query With Several OR Predicates Returns Wrong Results	2-8
2.1.13 EXCESS_TRANS Error After ROLLBACK of 2PC Transaction	2-9
2.1.14 Nested RDO FOR Loop Query Returns Wrong Results	2-10
2.1.15 Left Outer Join Query With a Table and a Subquery Returns Wrong Results	2-11
2.1.16 Excessive Snapshot Growth in 7.0.5	2-12
2.1.17 Switching to Match and Sort Strategy From Cross Strategy Causes Slow Down	2-13
2.1.18 Excessive Snapshot File Growth in 7.0.5	2-14
2.1.19 Query With OR Expression and Constant Predicates Returns Wrong Results	2-14
2.1.20 Query With Shared Expressions in OR Predicates Returns Wrong Results	2-16
2.1.21 Bugcheck at PSII2SCANRESETSCAN + 00000480	2-17
2.1.22 Query Joining Aggregate Subquery Returns Wrong Results	2-18

2.1.23	Ranked Index Bugcheck PSII2CHKPARCHDCARD + 0000000 Fixed	2-19
2.1.24	Control of Sort Work Memory Allocation	2-21
2.1.25	ORDER BY Query Joining a Table With a Subselect of UNION Legs Returns Wrong Order	2-21
2.1.26	Multiple Index Segments Not Used For OR Query	2-22
2.1.27	Outer Join Query Bugchecks When Its Equi-join Predicate is Transitive With a Subselect Query	2-23
2.1.28	Bugcheck at PSII2SPLITNODE + 3A8	2-24
2.1.29	Processes Vanished When Using Ranked Indexes	2-24
2.1.30	Query With COALESCE Function Returns Wrong Results	2-24
2.1.31	Query with DISTINCT, GROUP BY and ORDER BY Returns Wrong Results	2-25
2.1.32	Query With CAST Function on CURRENT_DATE Returns Wrong Results	2-25
2.1.33	Aggregate Query With Partitioned Index Bugchecks When SQL92 Dialect is Set	2-26
2.1.34	Complex Nested View Query Returns Wrong Results When Match Strategy is Used	2-27
2.2	SQL Errors Fixed	2-30
2.2.1	Unexpected Column Name Present in SQLDA Structure	2-30
2.2.2	Unexpected Errors from Date/Time Subtraction	2-30
2.2.3	SQL Functions Can Now Be Called From Triggers	2-32
2.2.4	Oracle LEVEL1 Dialect Not Displayed by SHOW CONNECTIONS ..	2-33
2.2.5	GET DIAGNOSTICS Option TRANSACTION_CHANGE_ALLOWED Errors	2-34
2.2.6	Temporary File Cleanup During Precompiled SQL	2-35
2.2.7	SET TRANSACTION Looses RESERVING Clause	2-35
2.2.8	Unexpected Informational Message Issued for Some Statements ..	2-37
2.2.9	Oracle RDBMS Style Join Syntax Does Not Support ALL Predicate	2-37
2.2.10	DROP STORAGE AREA ... CASCADE May Fail With ACCVIO Error	2-38
2.2.11	Exception in Nested Routines Did Not Close Index Scans	2-38
2.3	Oracle RMU Errors Fixed	2-39
2.3.1	/OPEN_MODE and /CLOSE_WAIT for RMU/RESTORE and /COPY	2-39
2.3.2	RMU Online Backup Consumes Excessive CPU Resources	2-40
2.3.3	SHOW STATS "Report" Does Not Include Logical Area "Graphs" ..	2-40
2.3.4	RMU/UNLOAD/AFTER_JOURNAL Sort Avoidance	2-40
2.3.5	RMU/UNLOAD/AFTER_JOURNAL Required All AIJ Backups to be Quiet-Point	2-40
2.3.6	RMU /BACKUP /AFTER_JOURNAL Delay Reduced	2-40
2.3.7	RMU/LOAD Starts Read-Only Transaction	2-41
2.3.8	RMU/UNLOAD/AFTER_JOURNAL Starts Read-Only Transaction ...	2-41
2.3.9	RMU/SHOW STATISTIC /OUTPUT and /RESET Display Wrong Statistics Values	2-41
2.3.10	RMU/EXTRACT Bugchecks During /ITEM=PROTECTION	2-41
2.3.11	RMU/SHOW STATISTICS /UNTIL Hangs	2-42
2.3.12	AIJ Quiet-Point Backup With Hot Standby and Row Cache Stops Checkpoint Advancement	2-42
2.3.13	Erroneous ABMBITERR Errors When Verifying Large Storage Areas	2-42

2.3.14	Long Transaction Logfile and "Checkpoint Information" Screen Have Same Threshold	2-43
2.3.15	RMU/UNLOAD/AFTER_JOURNAL Checks Callback Routine Return Status	2-43
2.3.16	RMU /RECOVER Fails When LogMiner Enabled	2-43
2.3.17	RMU /UNLOAD /AFTER_JOURNAL Output Records Larger Than 32767 Bytes	2-43
2.3.18	RMU /UNLOAD /AFTER_JOURNAL Enhanced VARCHAR Support	2-44
2.3.19	RMU OPTIMIZER_STATISTICS Leaves LOG File With Large Allocation	2-44
2.3.20	RMU /UNLOAD /AFTER_JOURNAL "/OPTIONS=SHARED_READ" Qualifier	2-45
2.3.21	RMU/COLLECT OPTIMIZER/READ_ONLY Bugcheck	2-45
2.3.22	RMU/LOAD Exception in READ_BLOB Routine if Empty Query Header	2-46
2.3.23	RMU/VERIFY/INDEX Always Used Two SORTWORK Files	2-46
2.3.24	RMU/ANALYZE/PLACE Gives 0 Path Length for Some Indexes	2-46
2.3.25	RMU/REPAIR/ABM/SPAM Causes Multiple AIP Page Pointers	2-47
2.3.26	RMU/RESTORE/OPEN_MODE Did Not Set Database Close Mode	2-47
2.3.27	RMU/SHOW STATISTICS Bad Results With /RESET/OUTPUT/CLUSTER	2-48
2.3.28	RMU/VERIFY Does Not Report Cause of VMS SORT Error	2-48
2.3.29	RMU/VERIFY RMU-E-BADSEG Typ Message Insufficient	2-49
2.3.30	RMU/CONVERT Multischema Database Problem	2-49
2.3.31	RMU/RESTORE From Multiple Tapes Gives False RMU-F-BACFILCOR	2-49
2.3.32	RMU/RECLAIM Set SPAM Thresholds Incorrectly on Uniform Areas	2-50
2.3.33	ACCVIOs from RMU/COPY/ONLINE or RMU/BACKUP/AFTER on OpenVMS Alpha V7.1	2-51
2.3.34	RMU/REPAIR/INIT=FREE_PAGES Corrupted Sorted RANKED Indexes	2-51
2.3.35	RMU Extract Did Not Process Empty QUERY HEADER Segments	2-52
2.4	Row Cache Errors Fixed	2-52
2.4.1	Row Cache Recovery Began at Too Old Checkpoint With Hot Standby	2-53
2.4.2	DBR Buffer Count for Node-Failure Recovery With Row Cache	2-53
2.5	Hot Standby Errors Fixed	2-53
2.5.1	RMU/REPLICATE AFTER_JOURNAL STOP/ABORT Should Close Database	2-53

3 Software Errors Fixed in Oracle Rdb7 Release 7.0.5

3.1	Software Errors Fixed That Apply to All Interfaces	3-1
3.1.1	Query Using Match Strategy Outline Returns Wrong Results	3-1
3.1.2	Database Recovery Process Bugchecks at DIOCCHDBR\$UNLATCH_GRCL + 00000398	3-2
3.1.3	Dynamic Optimizer Problem with Zigzag Match	3-2
3.1.4	DBR Bugcheck in DBR\$RECOVER_RCS Due to AIJ Related Database Shutdown	3-3
3.1.5	Bugchecks at DIOCCH\$FETCH_SNAP_SEG + 000005C4	3-3
3.1.6	Random Corrupt Pages on Fast Processors	3-3

3.1.7	Wrong Results from 3-Way Join Using Cross/Zigzag_Match	3-4
3.1.8	Query Slowdown Caused by Subquery With MIN/MAX Functions	3-4
3.1.9	GROUP BY/HAVING Query From a View With LIMIT TO Clause Returns Wrong Results	3-5
3.1.10	Query Returns Wrong Results When the Sequence of Same Context Predicates is Broken Up	3-5
3.1.11	Wrong Results When GROUP BY Columns are NOT Leading Subset of UNION Columns	3-6
3.1.12	New Index Scan Algorithm Not Effective With Some Sorted Indices	3-7
3.1.13	Wrong Results From a View Query With Left Outer Join and SUBSTRING Function	3-8
3.1.14	Query With EXISTS and SUBSTRING Bugchecks	3-9
3.1.15	Memory Leak for Trigger Actions	3-9
3.2	SQL Errors Fixed	3-10
3.2.1	Incorrect Output in SHOW STORAGE AREA (USAGE) Display	3-10
3.3	Oracle RMU Errors Fixed	3-10
3.3.1	RMU/BACKUP/AFTER/EDIT_FILE Keyword "YEAR" is Producing a Value of 1999	3-11
3.3.2	RMU/SHOW STATS "Average Per Transaction" is Relative to Epoch	3-11
3.4	Hot Standby Errors Fixed	3-12
3.4.1	Hot Standby Performance Impact on Master Database is Substantial	3-12

4 Software Errors Fixed in Oracle Rdb7 Release 7.0.4

4.1	Software Errors Fixed That Apply to All Interfaces	4-1
4.1.1	RMU/LOAD into Temporary Table	4-1
4.1.2	Divide by Zero Error in Query on Large Table	4-1
4.1.3	Wrong Results With COUNT DISTINCT CASE	4-2
4.1.4	Bugcheck at RDMS\$\$RDMSHEMA_UNLOAD_META+40 on Drop Area With Cascade	4-2
4.1.5	Unexpected I/O During DROP and TRUNCATE TABLE	4-3
4.1.6	Incorrect Rounding of Negative Numbers in the Round Function	4-4
4.1.7	Ignored Join Order Led to Poor Query Performance	4-4
4.1.8	GROUP BY Query on a Distinct Subquery Returns Wrong Results . . .	4-4
4.1.9	After Image Journal File Format Change	4-6
4.1.10	ORDER BY Ignored in Query With a Sub-select Statement	4-6
4.1.11	Query With Sort/Forward Scan Instead of Reverse Scan Slows Down	4-6
4.1.12	Query With Selection Predicates Over UNION Legs Returns Wrong Results	4-7
4.1.13	Left Outer Join View Query With CASE Statement Returns Wrong Results	4-8
4.1.14	Query Slower Using Cross Strategy and Outline Fails to Restore to Match	4-10
4.2	SQL Errors Fixed	4-11
4.2.1	Unexpected UNSDATASS Error Reported by SQL Precompiler and Module Language	4-11
4.2.2	SQL IMPORT No Longer Evaluates Table and Column Constraints	4-11
4.2.3	Unexpected INVACC_OUT_PARA Error Generated by CREATE MODULE	4-11

4.2.4	Changed Behavior for CAST of Date/Time Values With Seconds Field	4-12
4.2.5	SQL Rejects Queries Which Use Column Named VALUE	4-14
4.3	Oracle RMU Errors Fixed	4-14
4.3.1	RMU Extract Has Enhanced Extract of Conditional Expressions	4-14
4.3.2	RMU/REPLICATE AFTER START Command Fails on TCP/IP With Large Port Numbers	4-15
4.3.3	SHOW STATS Cannot Replay /OPTIONS=ROW_CACHE Input File	4-15
4.3.4	RMU/SHOW LOCKS Difficult to Identify Lock Conflict Culprit	4-16
4.3.5	RMU BACKUP to Tape Hung if Bad Checksum	4-18
4.3.6	RMU BACKUP to Tape Hung on QUIT Response to Wrong Label Message	4-19
4.3.7	RMU/REPAIR/INIT=FREE_PAGES/ABM Did Not Return an Error	4-19
4.3.8	Incorrect BADIDXREL Messages From Online RMU Verify	4-19
4.3.9	RMU VERIFY Did Not Find a .RDA File After an RMU MOVE	4-20
4.4	Row Cache Errors Fixed	4-20
4.4.1	Row Cache Server Operator Notification	4-20
4.4.2	Row Cache Did Not Avoid Certain Database Writes	4-21
4.4.3	RMU /CLOSE /WAIT Would Not Always Wait When Row Cache Enabled	4-21
4.5	Hot Standby Errors Fixed	4-21
4.5.1	RMU/REPLICATE AFTER START Command Fails Due to Lost AIJ Write	4-21

5 Documentation Corrections

5.1	Documentation Corrections	5-1
5.1.1	RMU /UNLOAD /AFTER_JOURNAL NULL Bit Vector Clarification	5-1
5.1.2	Location of Host Source File Generated by the SQL Precompilers	5-3
5.1.3	Suggestion to Increase GH_RSRVPGCNT Removed	5-4
5.1.4	Clarification of the DDLDONOTMIX Error Message	5-5
5.1.5	Compressed Sorted Index Entry Stored in Incorrect Storage Area	5-5
5.1.6	Partition Clause is Optional on CREATE STORAGE MAP	5-7
5.1.7	Oracle Rdb Logical Names	5-7
5.1.8	Waiting for Client Lock Message	5-8
5.1.9	Documentation Error in <i>Oracle Rdb7 Guide to Database Performance and Tuning</i>	5-9
5.1.10	SET FLAGS Option IGNORE_OUTLINE Not Available	5-10
5.1.11	SET FLAGS Option INTERNALS Not Described	5-10
5.1.12	Documentation for VALIDATE_ROUTINE Keyword for SET FLAGS	5-11
5.1.13	Documentation for Defining the RDBSERVER Logical Name	5-11
5.1.14	Undocumented SET Commands and Language Options	5-12
5.1.14.1	QUIET COMMIT Option	5-12
5.1.14.2	COMPOUND TRANSACTIONS Option	5-13
5.1.15	Undocumented Size Limit for Indexes with Keys Using Collating Sequences	5-13
5.1.16	Changes to RMU/REPLICATE AFTER/BUFFERS Command	5-14
5.1.17	Change in the Way RDMAIJ Server is Set Up in UCX	5-15
5.1.18	CREATE INDEX Supported for Hot Standby	5-15
5.1.19	Dynamic OR Optimization Formats	5-15

6 Known Problems and Restrictions

6.0.1	Behavior Change in 'With System Logical_Name Translation' Clause	6-1
6.0.2	Carry-Over Locks and NOWAIT Transactions Clarification	6-2
6.0.3	Strict Partitioning May Scan Extra Partitions	6-2
6.0.4	Exclusive Access Transactions May Deadlock With RCS Process	6-2
6.0.5	Oracle Rdb and OpenVMS ODS-5 Volumes	6-3
6.0.6	Clarification of the USER Impersonation Provided by the Oracle Rdb Server	6-3
6.0.7	Index STORE Clause WITH LIMIT OF Not Enforced in Single Partition Map	6-4
6.0.8	Unexpected NO_META_UPDATE Error Generated by DROP MODULE ... CASCADE When Attached by PATHNAME	6-4
6.0.9	Unexpected DATEEQLILL Error During IMPORT With CREATE INDEX or CREATE STORAGE MAP	6-5
6.0.10	Application and Oracle Rdb Both Using SYSSHIBER	6-5
6.0.11	IMPORT Unable to Import Some View Definitions	6-6
6.0.12	AIJSERVER Privileges	6-7
6.0.13	Lock Remastering and Hot Standby	6-7
6.0.14	RDB_SETUP Privilege Error	6-8
6.0.15	Starting Hot Standby on Restored Standby Database May Corrupt Database	6-8
6.0.16	Restriction on Compound Statement Nesting Levels	6-8
6.0.17	Back Up All AIJ Journals Before Performing a Hot Standby Switchover Operation	6-10
6.0.18	Concurrent DDL and Read-Only Transaction on the Same Table Not Compatible	6-10
6.0.19	Oracle Rdb and the SRM_CHECK Tool	6-10
6.0.20	Oracle RMU Checksum_Verification Qualifier	6-11
6.0.21	Do Not Use HYPERSORT with RMU/OPTIMIZE/AFTER_JOURNAL (Alpha)	6-12
6.0.22	Restriction on Using /NOONLINE with Hot Standby	6-12
6.0.23	SELECT Query May Bugcheck with PSII2SCANGETNEXTBBCDUPLICATE Error	6-12
6.0.24	DBAPack for Windows 3.1 is Deprecated	6-12
6.0.25	Determining Mode for SQL Non-Stored Procedures	6-13
6.0.26	DROP TABLE CASCADE Results in %RDB-E-NO_META_UPDATE Error	6-15
6.0.27	Bugcheck Dump Files with Exceptions at COSI_CHF_SIGNAL	6-16
6.0.28	Interruptions Possible when Using Multistatement or Stored Procedures	6-16
6.0.29	Row Cache Not Allowed on Standby Database While Hot Standby Replication Is Active	6-17
6.0.30	Hot Standby Replication Waits when Starting if Read-Only Transactions Running	6-18
6.0.31	Error when Using the SYS\$LIBRARY:SQL_FUNCTIONS70.SQL Oracle Functions Script	6-18
6.0.32	DEC C and Use of the /STANDARD Switch	6-18
6.0.33	Excessive Process Page Faults and Other Performance Considerations During Oracle Rdb Sorts	6-19
6.0.34	Performance Monitor Column Mislabeled	6-20
6.0.35	Restriction Using Backup Files Created Later than Oracle Rdb7 Release 7.0.1	6-20
6.0.36	RMU Backup Operations and Tape Drive Types	6-21

6.0.37	Use of Oracle Rdb from Shared Images	6-21
6.0.38	Interactive SQL Command Line Editor Rejects Eight-Bit Characters	6-21
6.0.39	Restriction Added for CREATE STORAGE MAP on Table with Data	6-22
6.0.40	ALTER DOMAIN...DROP DEFAULT Reports DEFVALUNS Error . . .	6-22
6.0.41	Oracle Rdb7 Workload Collection Can Stop Hot Standby Replication	6-23
6.0.42	RMU Convert Command and System Tables	6-24
6.0.43	Converting Single-File Databases	6-24
6.0.44	Restriction when Adding Storage Areas with Users Attached to Database	6-24
6.0.45	Restriction on Tape Usage for Digital UNIX V3.2	6-25
6.0.46	Support for Single-File Databases to be Dropped in a Future Release	6-25
6.0.47	DECdtm Log Stalls	6-25
6.0.48	Cannot Run Distributed Transactions on Systems with DECnet/OSI and OpenVMS Alpha Version 6.1 or OpenVMS VAX Version 6.0	6-26
6.0.49	Multiblock Page Writes May Require Restore Operation	6-26
6.0.50	Oracle Rdb7 Network Link Failure Does Not Allow DISCONNECT to Clean Up Transactions	6-27
6.0.51	Replication Option Copy Processes Do Not Process Database Pages Ahead of an Application	6-27
6.0.52	SQL Does Not Display Storage Map Definition After Cascading Delete of Storage Area	6-28
6.0.53	ARITH_EXCEPT or Incorrect Results Using LIKE IGNORE CASE	6-28
6.0.54	Different Methods of Limiting Returned Rows from Queries	6-28
6.0.55	Suggestions for Optimal Usage of the SHARED DATA DEFINITION Clause for Parallel Index Creation	6-29
6.0.56	Side Effect when Calling Stored Routines	6-31
6.0.57	Nested Correlated Subquery Outer References Incorrect	6-32
6.0.58	Considerations when Using Holdable Cursors	6-34
6.0.59	INCLUDE SQLDA2 Statement Is Not Supported for SQL Precompiler for PL/I in Oracle Rdb Release 5.0 or Higher	6-35
6.0.60	SQL Pascal Precompiler Processes ARRAY OF RECORD Declarations Incorrectly	6-35
6.0.61	RMU Parallel Backup Command Not Supported for Use with SLS . . .	6-36
6.0.62	Oracle RMU Commands Pause During Tape Rewind	6-36
6.0.63	TA90 and TA92 Tape Drives Are Not Supported on Digital UNIX	6-36
6.1	Oracle CDD/Repository Restrictions	6-36
6.1.1	Oracle CDD/Repository Compatibility with Oracle Rdb Features	6-36
6.1.2	Multischema Databases and CDD/Repository	6-38
6.1.3	Interaction of Oracle CDD/Repository Release 5.1 and Oracle RMU Privileges Access Control Lists	6-38
6.1.3.1	Installing the Corrected CDDSHR Images	6-40
6.1.3.2	CDD Conversion Procedure	6-40

7 Enhancements

7.1	Enhancements Provided in Oracle Rdb7 Release 7.0.6	7-1
7.1.1	RMU Show Statistics "Hot Standby Throughput" Screen	7-1
7.1.2	RMU/UNLOAD/AFTER_JOURNAL /STATISTICS Totals Information	7-2
7.1.3	RMU/SHOW STATISTIC "Row Cache Information" Cache Name Shortcut	7-3
7.1.4	RMU Command Line Help Updated	7-3
7.1.5	New /EXCLUDE_TABLES Qualifier for RMU/COLLECT OPTIMIZER	7-3
7.1.6	RMU Show Statistic Utility "Hot Standby Log" Facility	7-4
7.1.7	Impact of NUMBER OF RECOVERY BUFFERS on Row Cache DBR Performance	7-5
7.1.8	RMU Show Statistics "Row Cache Overview" Screen	7-6
7.1.9	SHOW STATS "Row cache (one field)" Display Configuration	7-8
7.1.10	LogMiner Now Supports SQL*Loader Output Files	7-9
7.1.11	RMU /UNLOAD /AFTER_JOURNAL "/FORMAT" Qualifier	7-10
7.1.12	New WORKLOAD Item for RMU/EXTRACT	7-11
7.1.13	RMU /UNLOAD /AFTER_JOURNAL "/INCLUDE = ACTION" Qualifier	7-12
7.1.14	SHOW STATS "Logical Area Access" Logfile	7-13
7.1.15	New OPTIMIZE Qualifier for RMU Unload	7-15
7.1.16	RCS Can Ignore Duplicate Checkpoint Requests	7-16
7.1.17	RMU /UNLOAD /AFTER_JOURNAL "/PARAMETER" Qualifier	7-16
7.1.18	New /CLOSE_WAIT Qualifier for RMU/RESTORE Command	7-17
7.1.19	New /EXCLUDE_TABLES Qualifier for RMU/COLLECT OPTIMIZER_STATISTICS	7-17
7.1.20	CREATE MODULE Now Supports External Routines	7-18
7.1.20.1	DESCRIPTION	7-18
7.1.20.2	USAGE NOTES	7-20
7.1.20.3	EXAMPLE	7-20
7.2	Enhancements Provided in Oracle Rdb7 Release 7.0.5	7-25
7.2.1	SHOW STATISTIC "Checkpoint Analysis" Screen	7-25
7.2.2	RMU Show Statistic "Online Analysis Logfile" Facility	7-28
7.2.3	New OPTIMIZE Clause DML Statements	7-29
7.2.4	New RMU /RECLAIM Command	7-29
7.2.5	New RMU /SERVER RECORD_CACHE CHECKPOINT Command	7-30
7.2.6	RCS Cycles TID Value at Checkpoint Completion	7-30
7.2.7	New Option for the GET DIAGNOSTICS Statement/HOT_STANDBY_MODE	7-30
7.2.8	New Option for the GET DIAGNOSTICS Statement/TRANSACTION_CHANGE_ALLOWED	7-31
7.2.9	New Hot Standby Logicals	7-32
7.3	Enhancements Provided in Oracle Rdb7 Release 7.0.4	7-33
7.3.1	Suggestion To Increase Field Size On RMU SHOW STATISTIC	7-33
7.3.2	SHOW STATS "Logical Area Overview" Enhancements	7-34
7.3.3	RCS Can Map All Caches at Database Open	7-35
7.3.4	Performance Enhancements When Number of Cluster Nodes is 1	7-35
7.3.5	New ROW LENGTH Default Calculated for CREATE CACHE	7-36
7.3.6	RMU /CHECKPOINT /WAIT /UNTIL	7-37
7.3.7	RMU Extract Supports New AUDIT_COMMENT Option	7-37
7.3.8	Revised Oracle Rdb for OpenVMS Client Kit	7-37
7.4	Enhancements Provided in Oracle Rdb7 Release 7.0.3.1	7-38

7.4.1	Per-Process Monitoring for SHOW STATS	7-38
7.4.2	New DOMAINS Option for RMU/EXTRACT	7-38
7.4.3	New NO REORGANIZE clause for ALTER STORAGE MAP	7-39
7.4.4	New Options for the GET DIAGNOSTICS Statement	7-41
7.4.5	RMU/SHOW Statistic OPCOM Message Tracking	7-42
7.4.6	New Restricted_Access Qualifier for RMU/LOAD	7-45
7.4.7	RDO EDT Editor on OpenVMS Alpha Now Available	7-45
7.4.8	New Options Added to SQL EXTRACT Function	7-45
7.5	Enhancements Provided in Oracle Rdb7 Release 7.0.2.1	7-46
7.5.1	New RMU Extract ITEM=REVOKE_ENTRY	7-46
7.5.2	New Data Type Support for DEC FORTRAN	7-47
7.5.3	New Data Type Support for DEC C	7-48
7.5.4	RMU/SHOW STATISTIC "TSNBLK Allocation" Screen	7-49
7.5.5	Scheduled AIJ Switch-overs	7-51
7.5.6	New Logicals for RMU/OPTIMIZE/AFTER_JOURNAL	7-53
7.6	Enhancements Provided in Oracle Rdb7 Release 7.0.2	7-53
7.6.1	Query Performance and Cartesian Limit	7-53
7.6.2	Named Query Outlines May Now Be Used by Triggers and Constraints	7-54
7.6.3	GRANT and REVOKE Support * for Object Names	7-55
7.6.4	New SET and SHOW DISPLAY Statements for Interactive SQL	7-56
7.6.5	SQL Now Supports DBKEY String Literals	7-59
7.6.6	RMU/SHOW STATISTIC "Logical Area" Statistics Consume Large Amounts of Memory	7-60
7.6.7	RMU/OPEN/STATISTICS, RMU/CLOSE/STATISTICS Enhancement	7-60
7.6.8	RCS Periodic Cache Validation	7-62
7.6.9	Hot Standby Support for Alternate Remote Nodename	7-62
7.7	Enhancements Provided in Oracle Rdb7 Release 7.0.1.3	7-63
7.7.1	Exceeding Complex Query Limit Generated %RDMS-F-MAX_CCTX Error	7-63
7.7.2	New Maximum Equivalent Class Limit for Complex Queries	7-63
7.7.3	Monitor Consumes Less Virtual Memory when Opening Databases with Global Buffers	7-63
7.7.4	Restrictions Lifted for Strict Partitioning	7-64
7.7.5	Date Subtraction	7-65
7.7.6	Default Node Size Now Displayed After Index Is Created	7-65
7.7.7	RUI Buffers in a Global Section When Row Cache is Enabled	7-66
7.7.8	Enhancements to Range Queries on SORTED Indexes	7-67
7.7.9	UPDATE STATISTICS Clause for ALTER TABLE Statement Implemented for /TYPE=NREL	7-69
7.7.10	Relaxed Privilege Checking for Temporary Tables	7-70
7.7.11	Improved Estimation for INDEX Column References	7-70
7.7.12	SQL92 Intermediate Level UNIQUE Constraint Available in Rdb7	7-72
7.7.13	Enhancements to DROP STORAGE AREA ... CASCADE	7-75
7.7.14	New SQL SET FLAGS Options	7-78
7.7.15	New ADD PARTITION Clause for ALTER INDEX	7-79
7.7.16	Enhancement to the SET TRANSACTION Statement	7-82
7.7.17	Computed Column Restriction Lifted for CREATE TRANSFER	7-84
7.7.18	Change In Functionality for RESTRICTED ACCESS Clause	7-85
7.7.19	SQL Expression Support for ORDER BY and GROUP BY Clauses	7-85
7.7.20	[No]Commit Qualifier Added to RMU/RESTORE Command	7-86
7.7.21	/WAIT Qualifier Added to RMU/OPEN Command	7-86

7.7.22	Limit the Number and Size of AIJ Initialization I/O Buffers	7-86
7.7.23	RMU/SHOW SYSTEM and RMU/SHOW USERS Now Include Elapsed Times	7-87
7.7.24	New Restricted_Access Qualifier for RMU/LOAD	7-87
7.7.25	New Qualifier for RMU/SHOW STATISTICS Command	7-87
7.7.26	RMU/SHOW STATISTICS "Automatic Screen Capture" Facility	7-88
7.7.27	RMU/SHOW STATISTIC "Logical Area Overview" Screen	7-89
7.7.28	RMU/SHOW STATISTICS "Summary Tx Statistics" Screen	7-92
7.7.29	RMU/SHOW STATISTICS "Recovery Information" Screen	7-94
7.8	Enhancements Provided In Oracle Rdb7 Release 7.0.1.2	7-96
7.8.1	Monitor Process Uses Less ENQLM	7-96
7.8.2	RDM\$TTB_HASH_SIZE Logical Name	7-96
7.8.3	New SQLSTATE Value	7-96
7.8.4	Planned Change in Behavior for the UNIQUE Predicate	7-97
7.8.5	UNION ALL and Derived Tables Allow up to 2000 Value Expressions	7-98
7.8.6	RMU/DUMP/AFTER Command /START and /END Qualifiers Improved	7-98
7.8.7	RMU/SHOW STATISTICS "Stall Message Logfile" Option Real Time Lock Information	7-99
7.8.8	RMU/SHOW STATISTICS Utility "Stall Messages Log" Displays Stall Duration Information	7-100
7.8.9	RMU/SHOW STATISTICS "User-Defined Events" Enhancements	7-101
7.8.10	Added Detail to RMU/SHOW STATISTICS "SPAM Fetches" Screen	7-104
7.8.11	RMU/SHOW STATISTICS Enhanced to Prevent Database Hangs	7-108
7.8.12	New SHOW STATISTICS Utility "AIJ Backup Activity" Screen	7-110
7.9	Enhancements Provided in Oracle Rdb7 Release 7.0.1.1	7-111
7.9.1	Virtual Memory Statistics No Longer Collected	7-111
7.9.2	New Logical Name RDM\$CREATE_LAREA_NOLOGGING	7-111
7.9.3	Online Creation of Storage Areas Performed In Parallel	7-115
7.9.4	Oracle7 Outer Join Syntax Support	7-117
7.9.5	RMU/SHOW STATISTIC "Transaction Recovery Duration Estimate" Screen	7-117
7.9.6	RMU/SHOW STATISTIC "File Overview" Sorting and Filtering Enhancements	7-119
7.9.7	RMU/SHOW STATISTIC Utility /OPTION=CONFIRM Qualifier	7-123
7.9.8	RMU/SHOW STATISTIC Utility Fast Incremental Backup Display	7-123
7.9.9	RMU/SHOW STATISTIC Utility "Page Information" Zoom Screen	7-124
7.9.10	RMU/SHOW STATISTIC "Logical Area" Menu Filter Option	7-127
7.9.11	RMU/SHOW STATISTIC "Stall Messages" Screen Allows Wildcards	7-127
7.9.12	CPU Time Displayed Correctly	7-127

8 LogMiner for Rdb

8.1	RMU Set Logminer Command	8-1
8.2	RMU Unload After_Journal Command	8-3
8.3	Restrictions and Limitations with LogMiner for Rdb	8-11
8.4	Information Returned by LogMiner for Rdb	8-12
8.5	Record Definition Prefix for LogMiner Fields	8-13
8.6	SQL Table Definition Prefix for LogMiner Fields	8-13
8.7	Segmented String Columns	8-14
8.8	Additional Examples	8-14

8.8.1	Example .rrd for the EMPLOYEES Table	8-14
8.8.2	Callback Module for the EMPLOYEES Table	8-15
8.8.3	Using LogMiner and the RMU Load Command to Replicate Table Data	8-16
8.8.4	Using LogMiner to Minimize Application Downtime for Maintenance	8-18
8.8.5	Using an OpenVMS Pipe	8-19

A Implementing Row Cache

A.1	Overview	A-1
A.1.1	Introduction	A-1
A.1.2	Database Functions Using Row Cache	A-2
A.1.3	Writing Modified Rows to Disk	A-3
A.1.4	Row Cache Checkpointing and Sweeping	A-4
A.1.5	Node and Process Failure Recovery	A-5
A.1.5.1	Process Failure	A-6
A.1.5.2	Node Failure	A-6
A.1.5.3	The RCS Process and Database Recovery	A-8
A.1.6	Considerations When Using the Row Cache Feature	A-8
A.2	Requirements for Using Row Caches	A-10
A.3	Designing and Creating a Row Cache	A-10
A.3.1	Reserving Slots for Row Caches	A-10
A.3.2	Row Cache Types	A-11
A.3.2.1	Assigning Storage Areas to Row Caches	A-12
A.3.2.2	Assigning Tables to Row Caches	A-12
A.3.3	Sizing a Row Cache	A-13
A.3.4	Choosing Memory Location	A-15
A.3.4.1	Sizing Considerations	A-18
A.4	Using Row Cache	A-19
A.4.1	Enabling and Disabling Row Cache	A-20
A.4.2	Specifying Checkpointing and Sweeping Options	A-20
A.4.2.1	Choosing the Checkpoint Source and Target Options	A-20
A.4.2.2	Choosing the Checkpoint Interval	A-22
A.4.2.3	Specifying Sweeping Parameters	A-22
A.4.2.4	Specifying the Size and Location of the Cache Backing File	A-23
A.4.3	Controlling What is Cached in Memory	A-24
A.4.3.1	Row Replacement Strategy	A-24
A.4.3.2	Inserting Rows into a Cache	A-24
A.5	Examining Row Cache Information	A-27
A.5.1	RMU Show Statistics Screens and Row Caching	A-31
A.6	Examples	A-32
A.6.1	Loading a Logical Area Cache	A-32
A.6.2	Caching Database Metadata	A-32
A.6.3	Caching a Sorted Index	A-34

B Row Cache Statements

B.1	ALTER DATABASE Statement	B-1
B.1.1	Overview	B-1
B.1.2	Environment	B-1
B.1.3	Format	B-2

B.1.4	Arguments	B-4
B.1.4.1	RECOVERY JOURNAL (BUFFER MEMORY IS {LOCAL GLOBAL})	B-4
B.1.4.2	RESERVE n CACHE SLOTS	B-4
B.1.4.3	CACHE USING row-cache-name	B-5
B.1.4.4	NO ROW CACHE	B-5
B.1.4.5	ROW CACHE IS ENABLED/ROW CACHE IS DISABLED	B-5
B.1.4.5.1	CHECKPOINT TIMED EVERY N SECONDS	B-5
B.1.4.5.2	CHECKPOINT ALL ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO DATABASE	B-6
B.1.4.5.3	LOCATION IS directory-spec	B-6
B.1.4.5.4	NO LOCATION	B-6
B.1.4.6	ADD CACHE clause	B-6
B.1.4.6.1	ALLOCATION IS n BLOCK/ALLOCATION IS n BLOCKS	B-6
B.1.4.6.2	CACHE SIZE IS n ROW/CACHE SIZE IS n ROWS	B-7
B.1.4.6.3	CHECKPOINT ALL ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO DATABASE	B-7
B.1.4.6.4	EXTENT IS n BLOCK/EXTENT IS n BLOCKS	B-7
B.1.4.6.5	LARGE MEMORY IS ENABLED/LARGE MEMORY IS DISABLED	B-7
B.1.4.6.6	LOCATION IS directory-spec	B-8
B.1.4.6.7	NO LOCATION	B-8
B.1.4.6.8	NUMBER OF RESERVED ROWS IS n	B-8
B.1.4.6.9	NUMBER OF SWEEP ROWS IS n	B-8
B.1.4.6.10	ROW LENGTH IS n BYTE/ROW LENGTH IS n BYTES	B-8
B.1.4.6.11	ROW REPLACEMENT IS ENABLED/ROW REPLACEMENT IS DISABLED	B-8
B.1.4.6.12	SHARED MEMORY IS SYSTEM/SHARED MEMORY IS PROCESS	B-9
B.1.4.6.13	WINDOW COUNT IS n	B-9
B.1.4.7	ALTER CACHE row-cache-name	B-9
B.1.4.7.1	row-cache-params	B-9
B.1.4.7.2	DROP CACHE row-cache-name CASCADE	B-9
B.1.4.7.3	DROP CACHE row-cache-name RESTRICT	B-9
B.2	CREATE DATABASE	B-9
B.2.1	Overview	B-9
B.2.2	Environment	B-10
B.2.3	Format	B-10
B.2.4	Arguments	B-12
B.2.4.1	RECOVERY JOURNAL (BUFFER MEMORY IS {LOCAL GLOBAL})	B-12
B.2.4.2	CACHE USING row-cache-name	B-12
B.2.4.2.1	NO ROW CACHE	B-13
B.2.4.3	RESERVE n CACHE SLOTS	B-13
B.2.4.4	ROW CACHE IS ENABLED/ROW CACHE IS DISABLED	B-13
B.2.4.4.1	CHECKPOINT TIMED EVERY N SECONDS	B-13
B.2.4.4.2	CHECKPOINT ALL ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO DATABASE	B-13
B.2.4.4.3	LOCATION IS directory-spec	B-14
B.2.4.4.4	NO LOCATION	B-14
B.3	CREATE CACHE Clause	B-14

B.3.1	Environment	B-14
B.3.2	Format	B-14
B.3.3	Arguments	B-15
B.3.3.0.1	CACHE row-cache-name	B-15
B.3.3.0.2	ALLOCATION IS n BLOCK/ALLOCATION IS n BLOCKS ...	B-15
B.3.3.0.3	EXTENT IS n BLOCK/EXTENT IS n BLOCKS	B-15
B.3.3.0.4	CACHE SIZE IS n ROW/CACHE SIZE IS n ROWS	B-15
B.3.3.0.5	CHECKPOINT ALL ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO DATABASE	B-16
B.3.3.0.6	LARGE MEMORY IS ENABLED/LARGE MEMORY IS DISABLED	B-16
B.3.3.0.7	ROW REPLACEMENT IS ENABLED/ROW REPLACEMENT IS DISABLED	B-16
B.3.3.0.8	LOCATION IS directory-spec	B-16
B.3.3.0.9	NO LOCATION	B-17
B.3.3.0.10	NUMBER OF RESERVED ROWS IS n	B-17
B.3.3.0.11	NUMBER OF SWEEP ROWS IS n	B-17
B.3.3.0.12	ROW LENGTH IS n BYTE/ROW LENGTH IS n BYTES	B-17
B.3.3.0.13	SHARED MEMORY IS SYSTEM/SHARED MEMORY IS PROCESS	B-17
B.3.3.0.14	WINDOW COUNT IS n	B-17
B.3.4	Usage Notes	B-18

C Release Notes Relating to the Row Cache Feature

C.1	Software Errors Fixed That Apply to All Interfaces	C-1
C.1.1	RCS Maximum Log File Size Control Logical	C-1
C.1.2	New RMU /SET ROW_CACHE [/ENABLE /DISABLE] Command	C-1
C.1.3	RCS Clearing "GRIC" Reference Counts	C-1
C.1.4	Row Cache RDC File Name Change	C-2
C.1.5	VLM or System Space Buffer Corruption	C-3
C.1.6	Invisible Row After Erase and Store With Row Cache	C-3
C.1.7	Overriding RCS Checkpoint Timer Interval	C-4
C.1.8	Refresh RCS Metadata Information	C-4
C.1.9	RCS ACCVIO When Checkpointing All Row Caches to Database	C-4

D Known Problems and Restrictions Relating to the Row Cache Feature

D.1	Known Problems and Restrictions	D-1
D.1.1	RMU Online Verification Operations and Row Cache	D-1
D.1.2	Limitation: Online RMU /VERIFY and Row Cache	D-1
D.1.3	Adding Row Caches Requires Exclusive Database Access	D-2
D.1.4	Conflicts When Caching Metadata and Executing Certain SQL Database Operations	D-2

E Logical Names Relating to the Row Cache Feature

E.1	RDM\$BIND_CKPT_FILE_SIZE	E-1
E.2	RDM\$BIND_CKPT_TIME	E-1
E.3	RDM\$BIND_DBR_UPDATE_RCACHE	E-1
E.4	RDM\$BIND_RCACHE_INSERT_ENABLED	E-1
E.5	RDM\$BIND_RCACHE_LATCH_SPIN_COUNT	E-1
E.6	RDM\$BIND_RCACHE_RCRL_COUNT	E-2
E.7	RDM\$BIND_RCS_BATCH_COUNT	E-2
E.8	RDM\$BIND_RCS_CARRYOVER_ENABLED	E-2
E.9	RDM\$BIND_RCS_CKPT_COLD_ONLY	E-2
E.10	RDM\$BIND_RCS_CKPT_BUFFER_CNT	E-2
E.11	RDM\$BIND_RCS_CKPT_TIME	E-2
E.12	RDM\$BIND_RCS_CLEAR_GRICS_DBR_CNT	E-2
E.13	RDM\$BIND_RCS_CREATION_IMMEDIATE	E-3
E.14	RDM\$BIND_RCS_KEEP_BACKING_FILES	E-3
E.15	RDM\$BIND_RCS_LOG_FILE	E-3
E.16	RDM\$BIND_RCS_LOG_HEADER	E-3
E.17	RDM\$BIND_RCS_LOG_REOPEN_SIZE	E-3
E.18	RDM\$BIND_RCS_LOG_REOPEN_SECS	E-3
E.19	RDM\$BIND_RCS_PRIORITY	E-3
E.20	RDM\$BIND_RCS_SWEEP_COUNT	E-4
E.21	RDM\$BIND_RCS_VALIDATE_SECS	E-4
E.22	RDM\$BIND_RUJ_GLOBAL_SECTION_ENABLED	E-4

Examples

7-1	Original Index Definition	7-81
7-2	Adding a Partition Before the Final Partition	7-82
7-3	Adding a New Final Partition	7-82
7-4	Adding a Partition Before the Final Partition	7-82
A-1	Sizing a Row Cache in a Global Section or System Space Buffer	A-19
A-2	Sizing a Row Cache in VLM	A-19
A-3	Sizing a Row Cache in Memory with VLM Enabled	A-19
A-4	Row Cache Parameters	A-28

Figures

7-1	RESERVING Clause	7-83
-----	------------------	------

Tables

5-1	Object Type Values	5-9
6-1	Oracle CDD/Repository Compatibility for Oracle Rdb Features	6-37
7-1	Supported FORTRAN Datatypes	7-48
7-2	Supported C Datatypes	7-49
7-3	Rdb Flag Keywords	7-78
7-4	Screen Fields	7-93
7-5	Screen Fields	7-95
7-6	“SPAM Access” Screen Fields	7-105
7-7	Recommended Quota Minimums	7-116

8-1	Output Fields	8-12
A-1	Memory Locations of Row Cache Objects	A-17
A-2	Checkpoint Target Options	A-21

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb7 Release 7.0.6. The notes describe changed and enhanced features; upgrade and compatibility information; new and existing software problems and restrictions; and software and documentation corrections. These release notes cover both Oracle Rdb7 for OpenVMS Alpha and Oracle Rdb7 for OpenVMS VAX, which are referred to by their abbreviated name, Oracle Rdb7.

Intended Audience

This manual is intended for use by all Oracle Rdb7 users. Read this manual before you install, upgrade, or use Oracle Rdb7 Release 7.0.6.

Document Structure

This manual consists of thirteen chapters:

Chapter 1	Describes how to install Oracle Rdb7 Release 7.0.6.
Chapter 2	Describes software errors corrected in Oracle Rdb7 Release 7.0.6.
Chapter 3	Describes software errors corrected in Oracle Rdb7 Release 7.0.5.
Chapter 4	Describes software errors corrected in Oracle Rdb7 Release 7.0.4.
Chapter 5	Provides information not currently available in the Oracle Rdb7 documentation set.
Chapter 6	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb7 Release 7.0.6.
Chapter 7	Describes enhancements introduced in Oracle Rdb7 Release 7.0.6 and all previous 7.0 Releases.
Chapter 8	Introduction to the new LogMiner for Oracle Rdb features available in Release 7.0.4 and beyond.
Appendix A	Describes the Row Cache feature and functionality which was added in Oracle Rdb7 Release 7.0.1.5.
Appendix B	Describes the Row Cache Statements available in Oracle Rdb7 Release 7.0.1.5 and beyond.
Appendix C	Describes software errors relating to the Row Cache feature that have been corrected in Oracle Rdb7 Release 7.0.1.5 and beyond.
Appendix D	Describes problems and restrictions relating to the Row Cache feature known to exist in Oracle Rdb7 Release 7.0.1.5 and beyond.
Appendix E	Describes the logical names relating specifically to the Row Cache feature that are available in Oracle Rdb7 Release 7.0.1.5 and beyond.

Installing Oracle Rdb7 Release 7.0.6

This software update is installed using the standard OpenVMS Install Utility.

1.1 Requirements

The following conditions must be met in order to install this software update:

- Oracle Rdb7 must be shutdown before you install this update kit. That is, the command file `SYSS$STARTUP:RMONSTOP(70).COM` should be executed before proceeding with this installation. If you have an OpenVMS cluster, you must shutdown all versions of Oracle Rdb7 on all nodes in the cluster before proceeding.
- The installation requires approximately 100,000 free blocks on your system disk for OpenVMS VAX systems; 200,000 blocks for OpenVMS Alpha systems.

1.2 Invoking VMSINSTAL

To start the installation procedure, invoke the VMSINSTAL command procedure:

```
@SYS$UPDATE:VMSINSTAL variant-name device-name OPTIONS N
```

variant-name

The variant names for the software update for Oracle Rdb7 Release 7.0.6 are:

- RDBSF070 for Oracle Rdb7 for OpenVMS VAX standard version.
- RDBASF070 for Oracle Rdb7 for OpenVMS Alpha standard version.
- RDBMVF070 for Oracle Rdb7 for OpenVMS VAX multiversion.
- RDBAMVF070 for Oracle Rdb7 for OpenVMS Alpha multiversion.

device-name

Use the name of the device on which the media is mounted.

- If the device is a disk drive, such as a CD-ROM reader, you also need to specify a directory. For CD-ROM distribution, the directory name is the same as the variant name. For example:

```
DKA400:[RDBSF070.KIT]
```

- If the device is a magnetic tape drive, you need to specify only the device name. For example:

```
MTA0:
```

OPTIONS N

This parameter prints the release notes.

The following example shows how to start the installation of the VAX standard kit on device MTA0: and print the release notes:

```
$ @SYS$UPDATE:VMSINSTAL RDBSF070 MTA0: OPTIONS N
```

1.3 Stopping the Installation

To stop the installation procedure at any time, press Ctrl/Y. When you press Ctrl/Y, the installation procedure deletes all files it has created up to that point and exits. You can then start the installation again.

If VMSINSTAL detects any problems during the installation, it notifies you and a prompt asks if you want to continue. You might want to continue the installation to see if any additional problems occur. However, the copy of Oracle Rdb7 installed will probably not be usable.

1.4 After Installing Oracle Rdb7

This update provides a new Oracle Rdb7 Oracle TRACE facility definition. Any Oracle TRACE selections that reference Oracle Rdb7 will need to be redefined to reflect the new facility version number for the updated Oracle Rdb7 facility definition, "RDBVMSV7.0-6".

If you have Oracle TRACE installed on your system and you would like to collect for Oracle Rdb7, you must insert the new Oracle Rdb7 facility definition included with this update kit.

The installation procedure inserts the Oracle Rdb7 facility definition into a library file called EPC\$FACILITY.TLB. To be able to collect Oracle Rdb7 event-data using Oracle TRACE, you must move this facility definition into the Oracle TRACE administration database. Perform the following steps:

1. Extract the definition from the facility library to a file (in this case, RDBVMS.EPC\$DEF).

```
$ LIBRARY /TEXT /EXTRACT=RDBVMSV7.0-6 -  
_$_ /OUT=RDBVMS.EPC$DEF SYS$SHARE:EPC$FACILITY.TLB
```
2. Insert the facility definition into the Oracle TRACE administration database.

```
$ COLLECT INSERT DEFINITION RDBVMS.EPC$DEF /REPLACE
```

Note that if you are installing the multiversion variant of Oracle Rdb7, the process executing the INSERT DEFINITION command must use the version of Oracle Rdb7 that matches the version used to create the Oracle TRACE administration database or the INSERT DEFINITION command will fail.

1.5 Alpha Wildfire Processor Support Added

As of this release of Rdb7, Oracle Rdb7 Release 7.0.6, the Alpha Wildfire processor is supported. See <http://metalink.oracle.com> for specifics on which Wildfire configurations are supported.

1.6 Maximum OpenVMS Version Check Added

As of Oracle Rdb7 Release 7.0.1.5, a maximum OpenVMS version check has been added to the product. Oracle Rdb has always had a minimum OpenVMS version requirement. With 7.0.1.5 and for all future Oracle Rdb releases, we have expanded this concept to include a maximum VMS version check and a maximum supported processor hardware check. The reason for this check is to improve product quality.

OpenVMS Version 7.2-n is the maximum supported version of OpenVMS.

As of Oracle Rdb7 Release 7.0.3, the Alpha EV6 processor is supported. As of Oracle Rdb7 Release 7.0.5, the Alpha EV67 processor is supported. As of Oracle Rdb7 Release 7.0.6, the Alpha Wildfire processor is supported (see <http://metalink.oracle.com> for specifics on which Wildfire configurations are supported).

The check for the OpenVMS operating system version and supported hardware platforms is performed both at installation time and at runtime. If either a non-certified version of OpenVMS or hardware platform is detected during installation, the installation will abort. If a non-certified version of OpenVMS or hardware platform is detected at runtime, Oracle Rdb will not start.

Software Errors Fixed in Oracle Rdb7 Release 7.0.6

This chapter describes software errors that are fixed by Oracle Rdb7 Release 7.0.6.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 False Reports of RDB-F-OBSOLETE_METADATA

Bugs 1396508, 1386447, 1365601

Prior to Rdb Version 7.0.4, it was possible for a process to use obsolete metadata without knowing that the metadata was obsolete and therefore corrupt a database. A code change was made in version 7.0.4 to prevent the problem but it can result in false reports of RDB-F-OBSOLETE_METADATA.

The undetected obsolete metadata problem occurred when a process read metadata from the snapshot file in a scenario like the following:

Process A	Process B
1.Starts a Read Only Transaction	
2.Does anything except access table T1. This process cannot have touched the table since it attached to the database.	
	1.Starts a Read/Write Transaction
	2.Drops an Index on table T1 (or creates an index on T1 or modifies the definition of T1.)
	3.Rdb acquires a table lock for table T1.
	4.Commits the transaction and releases the table lock.
3.Attempts to access table T1. Rdb acquires the table lock in PR because it has been released by Process B.	
4.Reads the old metadata information from snapshots since its transaction is older than Process B's transaction.	
5.Compiles a request that accesses T1 using the obsolete metadata.	

The result is that Process A's request is compiled with obsolete metadata. This problem was detected (rarely) in internal testing and has been infrequently reported by customers.

A correction for the problem was added to Oracle Rdb version 7.0.4. Unfortunately, this correction contained an error that resulted in false reports of obsolete metadata by all or most processes attached to the database when metadata changes have been made but the metadata viewed by other database attaches is not obsolete at all. For some customers, this false positive result has caused interruptions to processing but it has not produced incorrect results or corrupted databases. The workaround for the problem is to close and reopen the database.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.2 Bugcheck at RDMS\$\$COMPILE_INDEX_MAPS + 00000447

A problem during the creation of executable code for placement and retrieval of index keys into and from a partitioned index caused the following bugcheck.

```
***** Exception at 00377645 : RDMS$$COMPILE_INDEX_MAPS + 00000447
%COSI-F-BUGCHECK, internal consistency failure
```

This problem may be seen when the optimizer chooses to access a partitioned index in carrying out a query containing an "OR" that covers the same set of columns within the partitioned index.

The example below using MF_PERSONNEL shows the type of index and query that may be effected by this problem.

```
SQL> create unique index EMP_IDX
cont>   on employees (
cont>     LAST_NAME, FIRST_NAME, MIDDLE_INITIAL
cont>     asc)
cont>   type is SORTED
cont>   store
  cont>     using (LAST_NAME, FIRST_NAME, MIDDLE_INITIAL)
cont>     in EMPIDS_LOW
cont>     with limit of ('Dement', 'Larry', 'T')
cont>     in EMPIDS_MID
cont>     with limit of ('Myotte', 'Charles', 'K')
cont>     otherwise in EMPIDS_OVER;
SQL> select LAST_NAME, FIRST_NAME, MIDDLE_INITIAL
cont> from employees
cont> where ( LAST_NAME > 'Danzig' or LAST_NAME = 'Danzig')
cont>       and          (LAST_NAME <= 'Dement');
```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.3 Memory Leak in Lock Structures

Bug 1229610

In prior versions of Oracle Rdb, a small amount of virtual memory associated with "page owner" locks was not being properly released at database disconnect time. This would result in a small loss of virtual memory with every database attach/disconnect cycle.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.4 Query With MIN Function on a Partitioned Table Returns Wrong Results

Bug 1264262

The following query, which finds the minimum value of a column from a table loaded with data starting from 0 to 99999, returns wrong results.

```
select min(coll) from T1;
Aggregate          Index only retrieval of relation T1
  Index name  NDX_T1_DESC [0:0]    Max key lookup
90000
```

The following query returns correct results.

```
select min(coll), max(coll) from T1;
Aggregate          Index only retrieval of relation T1
  Index name  NDX_T1_DESC [0:0]
0 99999
```

The index NDX_T1_DESC is defined as follows:

```
create index NDX_T1_DESC on T1 (coll desc, col2)
store using (coll)
in lc11 with limit of (90000)
in lc12 with limit of (80000)
in lc13 with limit of (70000)
in lc14 with limit of (60000)
in lc15 with limit of (50000)
in lc16 with limit of (40000)
in lc17 with limit of (30000)
in lc18 with limit of (20000)
in lc19 with limit of (10000)
otherwise in lc20;
```

The key parts of this query which contributed to the situation leading to the error are these:

1. The MIN function is applied on a column of a partitioned table
2. The index is descending

The workaround is to replace the index with an ascending column, instead of descending.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.5 Monitor Bugchecks at MON\$DELETE_UNREFERENCED_GBL

Bug 1273381

If the user who started the monitor did not have a valid default directory when executing the RMU/MONITOR START command, the monitor could later bugcheck with an exception similar to the following:

```
***** Exception at 000D104C : MON$DELETE_UNREFERENCED_GBL + 00001CCC
%SYSTEM-F-ACCVIO, access violation, reason mask=04, virtual address=FFFFFFFFFC'
```

The monitor logfile would contain text like the following:

```
- database shutdown of "DEV:[DIR]FOO.RDB:1" is complete
- database shutdown of "DEV:[DIR]FOO.RDB:1" is complete
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DEV:[DIR]RDMMONBUG.DMP;
```

Note the two occurrences of the "database shutdown" messages.

An invalid default directory may either be a directory that does not exist, or a logical name that is defined as a search list. For example, SYSS\$STARTUP would be an invalid default directory.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.6 Recovery Operation Work File Allocation

During a DBR recovery roll-forward (typically this will only occur when the "fast commit" feature is enabled), the DBR process may need to allocate work files. In previous versions of Oracle Rdb, these work files were created without a specified allocation or extend size. This could cause the files to be frequently extended as they were being accessed. In turn, additional I/Os would occur during recovery.

This problem has been corrected in Oracle Rdb7 Release 7.0.6. Oracle Rdb now allocates DBR recovery work files with larger allocation and extension values and attempts to "tune" the values based on the use of previous work files during the recovery operation.

Note

The additional file size may cause recovery operations to require slightly more disk space. Make sure to reserve enough disk space for the recovery work files.

2.1.7 Query With Shared Expression in OR Predicate Returns Wrong Results

Bug 1301603

The following query should return 100 rows but instead it returns 0 rows.

```
set flags 'stra,detail';
Select
    O.CLIENT_ORDER_ITEM_ID_NO,
    O.CREATN_DATE,
    O.CLIENT_SUBSCN_ID_NO,
    O.CLIENT_ORDER_LIST_ID_NO
From
    CLIENT_FUND_AMT_ASGNMT      A,
    CLIENT_ORDER_ITEM          O
Where
    (
        O.CLIENT_ORDER_LIST_ID_NO < 5000000
    or
        ( O.CLIENT_ORDER_LIST_ID_NO = 5000000 and
          O.CLIENT_SUBSCN_ID_NO < 5000000 )
    or
        ( O.CLIENT_ORDER_LIST_ID_NO = 5000000 and
          O.CLIENT_SUBSCN_ID_NO = 5000000 and
          O.CREATN_DATE < '01-Jan-1996' ) )
    AND
    O.CLIENT_ORDER_LIST_ID_NO = 1000780
;
Tables:
    0 = CLIENT_FUND_AMT_ASGNMT
    1 = CLIENT_ORDER_ITEM
Cross block of 2 entries
Cross block entry 1
Conjunct: (((1.CLIENT_ORDER_LIST_ID_NO < 5000000) OR
            ((1.CLIENT_ORDER_LIST_ID_NO = 5000000) AND
             (1.CLIENT_SUBSCN_ID_NO < 5000000))) OR
            ((1.CLIENT_ORDER_LIST_ID_NO = 5000000) AND
             (1.CLIENT_SUBSCN_ID_NO = 5000000) AND
             (1.CREATN_DATE < <datatype:35>))) AND
            (1.CLIENT_ORDER_LIST_ID_NO = 1000780))
Index only retrieval of relation CLIENT_ORDER_ITEM
```

```

Index name COI_SORTED_11 [1:1,1:2,2:3]
Key: (1.CLIENT_ORDER_LIST_ID_NO = 1000780) AND
      (1.CLIENT_ORDER_LIST_ID_NO = 5000000) AND
      (1.CLIENT_SUBSCN_ID_NO < 5000000) AND
      (1.CLIENT_SUBSCN_ID_NO = 5000000) AND
      (1.CLIENT_ORDER_LIST_ID_NO = 5000000) AND
      (1.CREATN_DATE < <datatype:35>)
Cross block entry 2
Conjunct: ((1.CLIENT_ORDER_LIST_ID_NO = 5000000) OR
           (1.CLIENT_ORDER_LIST_ID_NO = 5000000))
Index only retrieval of relation CLIENT_FUND_AMT_ASGNMT
Index name CFAA_BATCH_2 [0:0]
0 rows selected

```

The problem is manifested by the dumper diagnosis displayed by the SQL SET FLAGS command with the key word 'STRATEGY, DETAIL', where the additional conjunct node is created for the predicate "O.CLIENT_ORDER_LIST_ID_NO = 5000000" twice for the wrong context A.CLIENT_FUND_AMT_ASGNMT in the second cross leg.

The key parts of this query which contributed to the situation leading to the error are these:

1. The query has several AND and OR predicates on the same table
2. The same equality predicate is used more than once in OR predicates

As a workaround, the query can be rewritten so that the predicate "O.CLIENT_ORDER_LIST_ID_NO = 5000000" will not appear twice. See the following example.

```

Where
(
  O.CLIENT_ORDER_LIST_ID_NO < 5000000
or
  ( O.CLIENT_ORDER_LIST_ID_NO = 5000000 and      <== used only once
    ( O.CLIENT_SUBSCN_ID_NO < 5000000 or
      ( O.CLIENT_SUBSCN_ID_NO = 5000000 and
        O.CREATN_DATE < '01-Jan-1996' ) ) )
) and

```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.8 Left Outer Join Query With UNIONed Sub-queries Returns Wrong Results

Bug 1321980

The following Left Outer Join query should return 1 row.

```

select hp.xxx_id, life.number
  from xxx hp LEFT OUTER JOIN
    (select yyy_id,
         yyy_number
     from yyy a
    union
    select zzz_id,
         zzz_number
     from zzz
    ) life (id, number)
  ON hp.xxx_id = life.id
where life.number IS NULL;
Tables:
  0 = XXX
  1 = YYY
  2 = ZZZ
Conjunct: MISSING (<based on field>0)
Cross block of 2 entries      (Left Outer Join)
Cross block entry 1
  Get      Retrieval sequentially of relation XXX
Cross block entry 2
  Conjunct: 0.XXX_ID = <based on field>0
Merge of 1 entries
  Merge block entry 1
  Reduce  Sort
Merge of 2 entries
  Merge block entry 1
  Conjunct: MISSING (1.YYY_NUMBER)
  Get      Retrieval sequentially of relation YYY
  Merge block entry 2
  Conjunct: MISSING (2.ZZZ_NUMBER)
  Get      Retrieval sequentially of relation ZZZ
HP.XXX_ID  LIFE.YYY_NUMBER
    1239             NULL
    3185             NULL
2 rows selected

```

The following Left Outer Join query should return 1 row.

```

select id, number
  from (
    select hp.xxx_id,
         life.yyy_number
    from xxx hp
      LEFT OUTER JOIN
    (select yyy_id,
         yyy_number
     from yyy a
    union
    select zzz_id,
         zzz_number
     from zzz
    ) life
      ON hp.xxx_id = life.yyy_id
    ) joint (id, number)
where joint.number IS NULL;
Tables:
  0 = XXX
  1 = YYY
  2 = ZZZ
Merge of 1 entries
Merge block entry 1
Conjunct: MISSING (1.YYY_NUMBER)
Cross block of 2 entries      (Left Outer Join)
Cross block entry 1
  Get      Retrieval sequentially of relation XXX

```

```

Cross block entry 2
  Conjunct: 0.XXX_ID = <based on field>0
  Merge of 1 entries
    Merge block entry 1
    Reduce Sort
  Merge of 2 entries
    Merge block entry 1
    Conjunct: MISSING (1.YYY_NUMBER)
    Get Retrieval sequentially of relation YYY
    Merge block entry 2
    Conjunct: MISSING (2.ZZZ_NUMBER)
    Get Retrieval sequentially of relation ZZZ
0 rows selected

```

The key parts of this query which contributed to the situation leading to the error are these:

1. Left outer join on a table and a derived table from a union of sub-queries
2. IS NULL predicate is tested on the other column of the derived table

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.9 SPAM Thresholds Incorrectly Set

Bug 1331010

It was possible for storage area pages to have their Space Area Management (SPAM) thresholds incorrectly set to "full"—that is, threshold 3—when the pages contained nothing but deleted line index entries. Deleted line index entries may be reclaimed and thus the page should not be marked as full. Symptoms of this problem may be poor performance when scanning tables stored in a uniform area or storage areas extending even though there are few rows stored in the table mapped to that storage area.

This problem would only occur when rows were constantly being inserted and deleted from a table within a single transaction. When storing data on a page, Oracle Rdb7 cannot reclaim space that was freed by a delete operation done in the same transaction. The space must be reserved in case a ROLLBACK is done. The space reserved is regarded as "locked free space". When a page got full of locked free space and the last row on the page was deleted, Oracle Rdb7 would neglect to update the SPAM threshold value to reflect the fact that the page no longer contained any data; the page would remain marked as full.

This problem has been corrected in Oracle Rdb7 Release 7.0.6. The SPAM page is now properly updated to show that a page is not full when the last row on the page is deleted.

2.1.10 ORDER BY Query With IN Clause Returns Wrong Order

Bug 1297796

The following query with IN clause and ORDER BY clause returns the results in the wrong order.

```

select DAT_KR, TABEL, BEWERKING from T1
  where tabel in (201,1)
     order by DAT_KR asc ;
Get      Retrieval by index of relation T1
  Index name  WIJZIGING_DB_AGENTSCHAP_S [0:0] Bool      Reverse Scan
  DAT_KR      TABEL  BEWERKING
  12-MAY-2000 09:56:31.87      201  M
  12-MAY-2000 09:56:31.75      201  M
2 rows selected

```

The key parts of this query which contributed to the situation leading to the error are these:

1. Out of the 3 selected columns, two come from the index
2. Query has ORDER BY clause on the 1st segment of the column
3. IN clause is referenced on the 2nd segment of the index

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.11 RDMS-F-READ_ONLY Error When Storage Area Mode Changed by Other Node

Bug 1337089

An undeserved RDMS-F-READ_ONLY error could be returned when the allowed access mode for a storage area was changed by another node in the cluster. For example:

1. Node 1 manually opens database via RMU/OPEN
2. Node 2 manually opens database via RMU/OPEN
3. Node 1 changes the access mode of a storage area to READ ONLY
4. Node 2 attempts to access the storage area

The above sequence of events could result in the following error:

```

SQL> set transaction reserving zsis_rt2 for shared read;
%RDMS-F-READ_ONLY, read-only area <area name> must be readied in RETRIEVAL
mode only

```

Subsequent attempts to access the storage area from the same attach would succeed.

The problem would occur because Oracle Rdb7 would neglect to consistently refresh the storage area parameters after a storage area was modified.

This problem has been corrected in Oracle Rdb7 Release 7.0.6. When a storage area's access mode has been altered by one node in a cluster, other nodes in the cluster will refresh their copies of the storage area parameters.

2.1.12 Join Query With Several OR Predicates Returns Wrong Results

Bug 1327038

The following query joining a derived table and a table with several OR predicates returns the wrong results.

```

att 'file shipping';
SELECT count(*)
  FROM SHIP C3,
       (select C2.SHIP_TYPE, C2.SHIP_RANGE
        from SHIP C2 )
       as C1 (SHIP_TYPE, SHIP_RANGE)
 WHERE
   C1.SHIP_TYPE = C3.SHIP_TYPE AND
   (C1.SHIP_RANGE > 0 AND
    C3.TONNAGE_CAPACITY = 0)
   OR C1.SHIP_RANGE < 0
   OR C1.SHIP_RANGE = 0
   ;
Tables:
  0 = SHIP
  1 = SHIP
Aggregate: (COUNT)
Cross block of 2 entries
Cross block entry 1
  Get      Retrieval sequentially of relation SHIP
Cross block entry 2
  Merge of 1 entries
    Merge block entry 1
      Conjunct: ((1.SHIP_RANGE > 0) OR (1.SHIP_RANGE < 0) OR (1.SHIP_RANGE = 0))
      Get      Retrieval sequentially of relation SHIP

          324
1 row selected

```

The key parts of this query which contributed to the situation leading to the error are these:

1. Query joins a derived table and a table with several ORs
2. WHERE clause is composed of the join predicates on a common column, and 2 or more OR predicates
3. All OR predicates contain the predicate testing the same column for "GTR", "LSS", and "EQL" to a zero value

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.13 EXCESS_TRANS Error After ROLLBACK of 2PC Transaction

Bug 689567

When an application repeatedly starts a two-phase READ/WRITE transaction involving both local and remote databases, does some work, and then rolls back the transaction, it is possible to receive an erroneous RDB\$_EXCESS_TRANS error from one of the subsequent attempts to start another transaction.

This is a timing related issue and only shows up in certain circumstances. It appears that this is most likely to happen when all of the following are true:

1. The application program loops very quickly from the ROLLBACK of one transaction to the START of another
2. Both local and remote databases are involved
3. Both local and remote systems use ALPHA CPUs
4. The application program (the "local" end) is running on a very fast processor and/or the remote processor has a load on it

The bug causing this problem has been present since Rdb Version 6.1, but is only showing up now as faster systems are deployed. The problem does not occur when the transactions are COMMITed, or if two-phase commit (DECdtm) is not involved.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

Caution: An application program which attempts to start a second two-phase transaction on a database before the first one is complete will see the second RDB\$START_TRANSACTION call hang until the first transaction finishes. Previously, the second call would return the RDB\$_EXCESS_TRANS status code. This happens because Rdb cannot reliably distinguish when a two-phase transaction is in the process of rolling back from when it is still being worked on.

2.1.14 Nested RDO FOR Loop Query Returns Wrong Results

Bug 1301225

The following nested RDO FOR loop query should return 0 rows:

```
define RDMS$SET_FLAGS "Strategy, Detail"
$ RDO
data file testdb

FOR TZ IN TZ
  FOR TY IN TY
    with TZ.COL1 <> 0 AND
    NOT ( ANY TK IN TK CROSS TS IN TS OVER JOIN_COL
          WITH TS.J_ID = TZ.J_ID )
    PRINT TZ.J_ID, TZ.COL3

  END-FOR
END-FOR
Tables:
  0 = TZ
Get      Retrieval sequentially of relation TZ
Tables:
  0 = TY
  1 = TK
  2 = TS
Cross block of 2 entries
  Cross block entry 1
    Conjunct: (subselect = 0)
    Aggregate-F1: (COUNT)
  Cross block of 2 entries
    Cross block entry 1
      Conjunct: (0 <> 0)
      Conjunct: (2.J_ID = "")
      Get      Retrieval sequentially of relation TS
    Cross block entry 2
      Conjunct: (2.JOIN_COL = 1.JOIN_COL)
      Get      Retrieval sequentially of relation TK
  Cross block entry 2
    Retrieval sequentially of relation TY
J_ID      COL3
91000000  100
```

From the strategy diagnosis dumped out by the optimizer, it is noticed that the conjunct is incorrectly generated for the predicate "TZ.COL1 <> 0" for the table TS.

The key parts of this query which contributed to the situation leading to the error are these:

1. Two nested RDO FOR loops
2. The outer WITH clause is placed inside the inner FOR loop
3. The inner FOR loop has an aggregate query of NOT ANY on two other tables using CROSS join
4. The inner WITH clause has a join predicate referencing the column of the table from the outer loop

As a workaround, place the outer WITH clause right after the outer FOR loop.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.15 Left Outer Join Query With a Table and a Subquery Returns Wrong Results

Bug 1343048

The following query returns the wrong results:

```
set flags 'strateg,detail';
select C1.PROJ_ID, C1.ANTRAG_ID, C2.PROJ_ID, C2.ANTRAG_ID
  from T0 as C1
     LEFT OUTER JOIN
     (SELECT A1.PROJ_ID, A1.ANTRAG_ID
      from T1 A1, T2 A2
      where A1.LAND_CODE = A2.LAND_CODE )
     as C2 (PROJ_ID, ANTRAG_ID )
     on (C1.PROJ_ID = C2.PROJ_ID AND C1.ANTRAG_ID = C2.ANTRAG_ID),
     T3 as C3
  where C1.BEI_ART_ID = C3.BEI_ART_ID
     and C3.STIP_FLAG = 'J';
```

Tables:

```
0 = T0
1 = T1
2 = T2
3 = T3
```

Cross block of 2 entries

Cross block entry 1

Conjunct: (3.STIP_FLAG = "J")

Get Retrieval sequentially of relation T3

Cross block entry 2

Cross block of 2 entries (Left Outer Join)

Cross block entry 1

Get Retrieval sequentially of relation T0

Cross block entry 2

Conjunct: ((0.PROJ_ID = <based on field>0) AND (0.ANTRAG_ID = <based on field>0))

Merge of 1 entries

Merge block entry 1

Cross block of 2 entries

Cross block entry 1

Conjunct: (0.BEI_ART_ID = 3.BEI_ART_ID)

Get Retrieval sequentially of relation T1

Cross block entry 2

Conjunct: (1.LAND_CODE = 2.LAND_CODE)

Get Retrieval sequentially of relation T2

C1.PROJ_ID	C1.ANTRAG_ID	C2.PROJ_ID	C2.ANTRAG_ID
174700	2	NULL	NULL
174700	2	NULL	NULL

2 rows selected

The conjunct "(0.BEI_ART_ID = 3.BEI_ART_ID)" is incorrectly generated on the table T1.

The key parts of this query which contributed to the situation leading to the error are these:

1. There is a table left outer join with a subselect cqry using ON clause
2. The left outer join is then joined with a second table
3. The WHERE predicate includes the join predicate using the column from the first table and the second table, and a second predicate of equality.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.16 Excessive Snapshot Growth in 7.0.5

If the "NUMBER OF CLUSTER NODES" was set to 1, then it was possible for Oracle Rdb7 to quit reclaiming pages in the snapshot files causing them to grow continuously. This problem only occurred in Oracle Rdb7 Release 7.0.5.

The problem occurred when a process executing a read-only transaction as its last transaction detached from the database and that process was the last one using a "TSN block". In that situation, the on-disk TSNBLK entry for that process would not be cleared and its last transaction information would remain in the TSNBLK; the in-memory copy of the TSNBLK would correctly show that the transaction was complete. If the database was then closed before that TSNBLK was used again, then the on-disk copy of the TSNBLK would contain the stale transaction information for the read-only transaction. When the database was opened again, update transactions would see the stale TSNBLK information and believe that the read-only transaction was still active, preventing them from reclaiming space in the snapshot files.

The problem with stale information being left in the TSNBLK can be demonstrated with the following script:

```
$ SQL
  CREATE DATABASE FILENAME FOO
    NUMBER OF CLUSTER NODES 1;
  DISCONNECT ALL;
  ATTACH 'FILENAME FOO';
  SET TRANSACTION READ WRITE;
  COMMIT;
  SET TRANSACTION READ ONLY;
  COMMIT;
  EXIT
$ RMU/DUMP/HEADER/OPTIONS=DEBUG/OUTPUT=FOO.TXT FOO
$ SEARCH FOO.TXT "SLOT["
  SLOT[1.] SIP TSN = 0:64, COMMIT_TSN = 0:0.
$
```

In the above example, the stale slot resides in the first TSNBLK of the database so it will be reused the next time a user attaches to the database. However, TSNBLKs that are only referenced when there are more than the usual amount of users in the database are more likely to have stale information left in them.

To workaround the problem, have more users attach to the database concurrently so that the TSNBLK entry with the stale information gets reused. Each of those users must issue a READ WRITE transaction prior to detaching from the database. This will clear out the stale entries until the next time the database is

closed, but it will not prevent the problem from occurring again. As long as the database remains open, the problem will not occur.

To completely prevent the problem from occurring, the following logical must be defined system wide:

```
$ DEFINE/SYSTEM RDM$BIND_ONE_TROOT 0
```

Note

Extreme care must be exercised when defining the above logical. No databases may be open on the system at the time the logical is defined and all databases on the system must be closed if the logical is deassigned. This logical disables optimizations used by Oracle Rdb7 when NUMBER OF CLUSTER NODES is set to 1, and can significantly degrade database performance.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.17 Switching to Match and Sort Strategy From Cross Strategy Causes Slow Down

Bug 1168583

The following query switches to using match with sort strategy instead of cross strategy with 2 segment index scan and therefore slows down the performance of the query:

```
SELECT count(*)
FROM T1 cm left outer join T2 cx
      on cx.office_code = cm.office_code
WHERE cm.client_code = 'FOO' and
      cx.effective_date >= '01-oct-1999' and
      cx.effective_date <= '31-oct-1999';
```

Oracle Rdb chooses the match/sort strategy which is cheaper than the cross, but if the selectivity factors for selection predicates become more aggressive, the strategy will switch back to cross.

A new value 'Selectivity' has been added to the SQL SET FLAGS statement and the logical RDMS\$SET_FLAGS, to allow the Rdb optimizer to apply a set of predefined selectivity factors which are more aggressive in selectivity costing.

For example, for a non-equality predicate like "effective_date >= '01-OCT-1999'", Oracle Rdb uses rather conservative estimates of 0.35 (35% or 35 matches out of 100 items). In contrast, Oracle RDBMS uses 0.05 for such predicates. With the new SQL flag being set to 'Selectivity', Oracle Rdb will now use 0.1 (10%) for such predicates.

The key parts of this query which contributed to the situation leading to the error are these:

1. Query with join predicates and equality predicates
2. Query with non-equality predicates
3. Query with MISSING, CONTAINING, STARTS WITH, LIKE, and NOT predicates

As a workaround, sometimes running RMU/COLLECT OPTIMIZER may solve the problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.18 Excessive Snapshot File Growth in 7.0.5

Bug 1364592

In a cluster environment where some nodes were actively updating the database while other nodes were only occasionally updating the database, it was possible for snapshot pages to not be reclaimed. This problem only exists in Oracle Rdb7 Release 7.0.5.

The problem would happen when there was one or more nodes that would occasionally execute a short duration update transaction; the database would remain open between transactions. In that situation, processes on other nodes in the cluster would not always notice that the short duration update transaction was complete and would believe that the transaction was still active. This would prevent snapshot pages in the snapshot files from being reused and could lead to constant extensions of the snapshot files.

To workaroud the problem, keep an update transaction active on the nodes that normally don't do many transactions. For example:

```
SQL> ATTACH 'FILENAME MYDB';
SQL>
SQL> CREATE FUNCTION LIB$WAIT (IN :PARAM1 REAL)
cont>     RETURNS INT;
cont>     EXTERNAL NAME LIB$WAIT LOCATION 'SYS$SHARE:LIBRTL.EXE'
cont>     LANGUAGE GENERAL
cont>     GENERAL PARAMETER STYLE;
SQL>
SQL> CREATE TABLE DUMMY_TABLE (DUMMY_FIELD INT);
SQL> COMMIT;
SQL>
SQL> BEGIN
cont> -- Loop forever.
cont> LOOP
cont>     SET TRANSACTION READ WRITE;
cont>     -- The transaction must alter the database in some way.
cont>     -- Wait for one minute and then end the transaction.
cont>     INSERT INTO DUMMY_TABLE VALUES (LIB$WAIT (60));
cont>     ROLLBACK;
cont> END LOOP;
cont> END;
```

The above workaroud will only reliably work if the nodes that don't often update the database have less than 28 users attached to the database.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.19 Query With OR Expression and Constant Predicates Returns Wrong Results

Bug 1369801

The following query with OR expression and constant predicates returns wrong results:

```

set flags 'strategy,detail';
select t2.id, t1.data1, t1.data2, t1.pid
  from lar_addr_rec t1, left_oj_view t2
   where t2.id = t1.id and
        ((0=1 and t2.id = 123) OR
         (0=0 and t1.data1 = 0) OR
         1=2) and
        t1.data2 = '20412345678' and
        t1.pid = 100;
Tables:
  0 = LAR_ADDR_REC
  1 = LAR_REC      =
  2 = LAR_ADDR_REC =
  3 = LAR_REC      =====> from left_oj_view
  4 = LAR_ADDR_REC =
  5 = LINK_REC     =
Cross block of 2 entries
Cross block entry 1
  Conjunct: (0.data2 = '20412345678')
  Conjunct: (((0 = 1) OR (1 = 2)) AND          <=== should not be here
             (0.pid = 100))
  Get      Retrieval sequentially of relation LAR_ADDR_REC
Cross block entry 2
  Conjunct: ((<based on field>0 = 0.id) AND
             (((0 = 1) AND (<based on field>0 = 123)) OR
              (0 = 0) OR (1 = 2)))          <=== should not be here
Cross block of 2 entries      (Left Outer Join)
Cross block entry 1
  Reduce Sort
  Conjunct: (((<based on field>0 = 0.id) AND
             (((0 = 1) AND (<based on field>0 = 123)) OR
              ((0 = 0) AND (0.data1 = 0)) OR
              (1 = 2))))
Merge of 2 entries
Merge block entry 1
Cross block of 2 entries
Cross block entry 1
  Get      Retrieval sequentially of relation LAR_REC
Cross block entry 2
  Conjunct: ((1.id = 2.id) AND (2.data1 = 0))
  Get      Retrieval sequentially of relation LAR_ADDR_REC
Merge block entry 2
  Conjunct: ((MISSING (4.data1)) OR (4.data1 = 1))
Cross block of 2 entries      (Left Outer Join)
Cross block entry 1
  Get      Retrieval sequentially of relation LAR_REC
Cross block entry 2
  Get      Retrieval sequentially of relation LAR_ADDR_REC
Cross block entry 2
  Conjunct: (((<based on field>0 = 0.id) AND
             (((0 = 1) AND          <=== should not be here
              (<based on field>0 = 123))
              OR (0 = 0) OR (1 = 2)))) <=== should not be here
  Conjunct: (<based on field>0 = 5.id)
  Get      Retrieval sequentially of relation LINK_REC
0 rows selected

```

The constant predicates "0 = 1", "0 = 0", and "1 = 2" are incorrectly generated. (See the arrow <=== above).

The key parts of this query which contributed to the situation leading to the error are these:

1. A table join with a left outer join view, where another table is joined with another nested left outer join view with UNION clause

- The WHERE predicate contains join predicates and OR expressions with constant predicates, e.g. 0=1, 1=1, etc.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.20 Query With Shared Expressions in OR Predicates Returns Wrong Results

Bug 1367855

The following query with shared expressions in OR predicates returns wrong results; it should return 0 rows.

```
SET FLAGS 'strategy,detail';

SELECT t1.fund,
       t1.txn_no,
       t1.txn_type,
       t3.flag,
       t2.sch_type
FROM switch t1,
     sch_param t2,
     fund_acct t3
WHERE
  (t1.fund = t3.fund AND
   t1.txn_no = 1 AND
   t3.flag = '0' AND
   t1.txn_type = 'n'
   ) OR
  (t1.fund = t3.fund AND
   t1.txn_no = 1 AND
   t3.flag = '0' AND
   t1.txn_type = 'p' AND
   t2.sch_type = '1' AND
   NOT EXISTS (SELECT * FROM txn t4
               WHERE t4.fund = t1.fund)
   ) ;
```

Tables:

```
0 = SWITCH
1 = SCH_PARAM
2 = FUND_ACCT
3 = TXN
```

Cross block of 4 entries

Cross block entry 1

Conjunct: (0.TXN_NO = 1)

Get Retrieval sequentially of relation SWITCH

Cross block entry 2

Aggregate-F1: (COUNT-ANY)

Conjunct: (3.FUND = 0.FUND)

Get Retrieval sequentially of relation TXN

Cross block entry 3

Conjunct: (2.FLAG = '0')

Conjunct: ((0.TXN_TYPE = 'N') OR (0.TXN_TYPE = 'P')) AND <== wrong context
(0.FUND = 2.FUND) AND
(0.TXN_NO = 1) <== wrong context

Get Retrieval by index of relation fund_acct

Index name FUND_INV_ACCOUNT_NDX [1:1]

Key: (0.FUND = 2.FUND)

Cross block entry 4

Conjunct: ((1.SCH_TYPE = '1') AND (subselect = 0) AND
(0.FUND = 2.FUND) AND
(0.TXN_NO = 1) AND <== wrong context
(2.FLAG = '0')) <== wrong context

Get Retrieval sequentially of relation SCH_PARAM

```

T1.FUND   T1.TXN_NO   T1.TXN_TYPE   T3.FLAG   T2.SCH_TYPE
AAACCC           1   P             0         0
1 row selected

```

The key parts of this query which contributed to the situation leading to the error are these:

1. A query joining 3 tables
2. The WHERE predicate contains OR predicates with duplicated expressions in each OR leg
3. One of the OR legs has a subselect query with NOT EXISTS clause

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.21 Bugcheck at PSII2SCANRESETSCAN + 00000480

Bug 1073555

During the processing of multiple cursor streams within the same process, accessing the same sorted ranked index, a bugcheck may occur due to a problem with re-establishing addressing space after an index entry conflict.

The following is an example of the start of the call stack that may be seen in the bugcheck dump arising from this problem:

```

**** Exception at 00E49318 : PSII2SCANRESETSCAN + 00000480
Saved PC = 00E40758 : PSII2SCANGETNEXTBBCDUPLICATE + 000000A8
Saved PC = 00FF4548 : RDMS$$KOD_ISCAN_GET_NEXT + 00000820
...

```

This problem may occur whenever two cursor streams within the same process are reading the same sorted ranked index and are currently accessing the same index entry key containing multiple duplicates.

The number of duplicates in the entry must be sufficient to require an overflow node to be created for that entry and, additionally, one of the streams must be doing a scan of the index while the other stream either inserts or deletes a duplicate entry within the same index key.

Whenever an updater inserts or deletes a duplicate entry from a sorted ranked index duplicate bitmap, a concurrent (ie, within the same process) reader of the entry has to re-establish its context to ensure that the bitmap it is scanning is correctly updated to reflect the updater changes.

The process of re-establishing the scan context by the reader had a problem which resulted in the incorrect determination of the end of the current scan criteria which, in turn, caused an intentional bugcheck to be generated.

A workaround for this problem is to use a normal sorted index instead of a sorted ranked index.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.22 Query Joining Aggregate Subquery Returns Wrong Results

Bug 1359357

The following query, joining an aggregate subquery with an aggregate column in an equality predicate, returns wrong results; an extra row of t1.test_header_number (of value "1217504") is returned by the equality predicate "cast(t1.status as integer) = cnt.ct".

```
set flags 'strategy,detail';
select distinct t1.test_header_number, t1.status, cnt.ct
  from rept_test_head th join rept_test_values t2
    on t1.test_header_number = t2.test_header_number
  join
    ( select c.test_header_number, count(*)
      from
        rept_test_head p join
        rept_test_values c
          on p.test_header_number = c.test_header_number
      group by
        c.test_header_number ) AS cnt ( test_header_number, ct )
    on t1.test_header_number = cnt.test_header_number
  where
    t1.transflag = 'I' and
    t1.transtatus = ' ' and
    cast( t1.status as integer ) = cnt.ct ;
```

Tables:

- 0 = REPT_TEST_HEAD
- 1 = REPT_TEST_VALUES
- 2 = REPT_TEST_HEAD
- 3 = REPT_TEST_VALUES

Reduce Sort

Cross block of 3 entries

Cross block entry 1

- Conjunct: (0.TRANSFLAG = 'I')
- Conjunct: (0.TRANSTATUS = ' ')
- Get Retrieval sequentially of relation REPT_TEST_HEAD

Cross block entry 2

- Conjunct: (0.TEST_HEADER_NUMBER = 1.TEST_HEADER_NUMBER)
- Get Retrieval sequentially of relation REPT_TEST_VALUES

Cross block entry 3

- Conjunct: (CAST (0.STATUS AS INT) = <based on field>0) <== Missing code

Merge of 1 entries

Merge block entry 1

- Aggregate: (COUNT) Sort

Cross block of 2 entries

Cross block entry 1

- Get Retrieval sequentially of relation REPT_TEST_HEAD

Cross block entry 2

- Conjunct: (2.TEST_HEADER_NUMBER = 3.TEST_HEADER_NUMBER)
- Get Retrieval sequentially of relation REPT_TEST_VALUES

T1.TEST_HEADER_NUMBER	T1.STATUS	CNT.CT	
1217504	4	4	<== should not return this
1217505	2	2	
1217506	3	3	

...etc...

The key parts of this query which contributed to the situation leading to the error are these:

1. Query contains a subquery join with distinct, count and group aggregates
2. Query contains several equality predicates, where one of the predicates refers to the column of the subquery with count aggregate

As a workaround, the query works if the predicates are duplicated with an OR clause as in the following example.

```
where
  (t1.transflag = 'I' and
   t1.transtatus = ' ' and
   cast( t1.status as integer ) = cnt.ct )
OR
  (t1.transflag = 'I' and
   t1.transtatus = ' ' and
   cast( t1.status as integer ) = cnt.ct ) ;
```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.23 Ranked Index Bugcheck PSII2CHKPARCHDCARD + 0000000 Fixed

A problem in how index entries are converted from a duplicate to an unique entry may cause problems with accessing entries found after the corrupt entry in the same index node.

This problem pertains to sorted ranked indexes only.

This problem can occur when a large number of duplicates are added to the same index entry, causing the creation of an overflow node, and then they are subsequently removed.

Bugcheck dumps of this problem usually show the following call stack:

```
***** PSII2CHKPARCHDCARD + 00000002
RDMS$$KOD_ISCAN_GET_NEXT + 000002F3
RDMS$$EXE_NEXT + 00000643
. . .
or:
***** PSII2SCANGETNEXTUNIQUE + 0000027C
RDMS$$KOD_ISCAN_GET_NEXT + 000007B8
RDMS$$EXE_LEAF + 00001A28
RDMS$$EXE_NEXT + 00001E58
. . .
```

RMU/VERIFY of such a corrupted index node will show verify errors similar to the following:

```
%RMU-I-BGNNDXVER, beginning verification of index INDEX1
%RMU-I-BTRENTLEN, B-tree node entry 1 has an invalid data length of 8.
%RMU-I-BTRNODDBK, Dbkey of B-tree node is 217:26:0
%RMU-I-BTRERPATH, parent B-tree node of 217:26:0 is at 217:8:0
%RMU-I-BTRENTLEN, B-tree node entry 1 has an invalid data length of 8.
%RMU-I-BTRNODDBK, Dbkey of B-tree node is 217:24:1
%RMU-I-BTRENTLEN, B-tree node entry 2 has an invalid data length of 20992.
%RMU-I-BTRNODDBK, Dbkey of B-tree node is 217:24:1
%RMU-I-BTRPFXERR, area RDB$SYSTEM, page -1, line 48
storage record RANKED B-TREE NODE, with prefix key "NC11
contains a bad prefix key len, expected: 60, found: 65
%RMU-I-BTRPFXERR, area RDB$SYSTEM, page -1, line 48
storage record RANKED B-TREE NODE, with prefix key "NC11
contains a bad prefix key len, expected: 60, found: 65
%RMU-W-BTRLENERR, area RDB$SYSTEM, page 24, line 1
storage record RANKED B-TREE NODE, b-tree node length error
expected node length 269, found: 21082
%RMU-I-BTRHEACAR, Sum of entry cardinalities given as 4 ; expected 50
%RMU-I-BTRNODDBK, Dbkey of B-tree node is 217:24:1
%RMU-I-BTRERPATH, parent B-tree node of 217:24:1 is at 217:8:0
```

The following scenario describes how this problem occurs:

1. A number of duplicates are entered for the same entry causing a duplicate overflow node to be created.
2. All but one of the duplicate entries within the overflow node are subsequently removed.
3. All but one of the duplicate entries within the primary segment for the entry (the bitmap segment that is held within the entry inside the original index leaf node) are subsequently removed leaving one overflow node still containing one duplicate dbkey for this entry.
4. A subsequent deletion of the duplicate dbkey contained within the primary segment leads to the alteration of this entry to a unique entry.
5. The change to a unique entry corrupts the subsequent entry within that index node.

An example follows. A dump of an index node with this problem will show that one or more entries are corrupt with the following possible problems:

- Invalid prefix length
- Invalid cardinality fields
- Corrupt values in stored key
- Length on entry is incorrect causing 'unused length' errors

The following is an extract from a RMU/DUMP/LAREA of an index that has incorrectly stored duplicate dbkeys.

```

.... total B-tree node size: 430
00D9 200D F424 line 1 (217:24:1) index: set 217
0122 FFFFFFFF FFFF F428 owner 290:-1:-1
010D F430 269 bytes of entries
0200 F432 level 1, full suffix
06 00 3C 0049 F434 60 bytes stored, 0 byte prefix
20202020204E00390031310043004E00 F439 key '.N.C.11.9.N '
202020202020202020202020202020 F449 key ' '
:::: (1 duplicate line)
2020202020202020202020202020 F469 key ' '
2FA5 10 F475 pointer 290:761:4
0001 F478 entry cardinality 1.
0000 F47A leaf cardinality 0.
00 41 00 5207 F482 0 bytes stored, 65 byte prefix
20202020204E00390031310043004E00 pfx '.N.C.11.9.N '
202020202020202020202020202020 pfx ' '
(1 duplicate line)
000000002020202020202020202020 pfx ' ....'
60 pfx ''
0031 30 F487 pointer 290:-1:48
0031 F48A entry cardinality 49.
2059 F48C leaf cardinality 8281.
unused length of -20684 is invalid
00D9 00000009 0005 F5C2 next node: 217:9:5
00000000000000004 F5CA Sum Card '.....'

```

The only workaround for this problem is to drop and recreate the index as a normal non-ranked index.

Dropping and recreating the index as a ranked index will remove this problem from the index at the time, however, depending on insertion and removals of duplicates. the problem may re-occur.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.24 Control of Sort Work Memory Allocation

Oracle Rdb uses a built-in SORT32 package to perform many sort operations. Sometimes these sorts exhibit a significant performance problem when initializing work memory to be used for the sort. This behavior can be experienced, for example, when a very large sort cardinality is estimated but the actual sort cardinality is small.

In rare cases, it may be desirable to artificially limit the sort package's use of work memory. Two logicals have been created to allow this control. In general, there should be no need to use either of these logicals and misuse of them can significantly impact sort performance. Oracle recommends that these logicals be utilized carefully and sparingly.

The logical names are:

- **RDMS\$BIND_SORT_MEMORY_WS_FACTOR**
Specifies a percentage of the process's working set limit to be used when allocating sort memory for the built-in SORT32 package. If not defined, the default value is 75 (representing 75%), the maximum value is 75 (representing 75%), and the minimum value is 2 (representing 2%). Processes with very large working set limits can sometimes experience significant page faulting and CPU consumption while initializing sort memory. This logical name can restrict the sort work memory to a percentage of the process's maximum working set.
- **RDMS\$BIND_SORT_MEMORY_MAX_BYTES**
Specifies an absolute limit to be used when allocating sort memory for the built-in SORT32 package. If not defined, the default value is unlimited (up to 1GB), the maximum value is 2,147,483,647 and the minimum value is 32,768.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.25 ORDER BY Query Joining a Table With a Subselect of UNION Legs Returns Wrong Order

Bug 1381043

The following ORDER BY query joining a table and a subselect of UNION legs returns the wrong order:

```

set flags 'strategy,detail';
select coll from departments dept,
  (select employee_id from employees where employee_id = dept.manager_id
   union
   select employee_id from employees where employee_id = dept.manager_id
  ) as unionemp(coll)
order by coll;
Tables:
  0 = DEPARTMENTS
  1 = EMPLOYEES
  2 = EMPLOYEES
Cross block of 2 entries
Cross block entry 1
  Get      Retrieval by index of relation DEPARTMENTS
          Index name  DEPARTMENTS_INDEX [0:0]
Cross block entry 2
Merge of 1 entries
  Merge block entry 1
  Reduce  Sort
Merge of 2 entries
  Merge block entry 1
  Index only retrieval of relation EMPLOYEES
          Index name  EMPLOYEES_HASH [1:1]      Direct lookup
          Key: (1.EMPLOYEE_ID = 0.MANAGER_ID)
  Merge block entry 2
  Index only retrieval of relation EMPLOYEES
          Index name  EMPLOYEES_HASH [1:1]      Direct lookup
          Key: (2.EMPLOYEE_ID = 0.MANAGER_ID)

```

The problem exists in a similar query with DISTINCT instead of UNION, as follows:

```

select coll from departments dept,
  (select distinct employee_id from employees
   where employee_id = dept.manager_id
  ) as dist_emp(coll)
order by coll;

```

The key parts of this query which contributed to the situation leading to the error are these:

1. The query joins a table with a subselect of UNION legs with an ORDER BY clause on the same column as the unioned column, OR
2. The query joins a table with a subselect of DISTINCT with an ORDER BY clause on the same column as the distinct column

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.26 Multiple Index Segments Not Used For OR Query

Bug 1230788

A problem was reported where a query was taking over two minutes to execute. This happened using Oracle Rdb V6.1-12 and Oracle Rdb7 as well. A simplified version of the query follows:

```

select la_number, priority, location_number, extended_ws_required
from mic_requested_tests
where worksheet_code = 'WBLOOD' and
      (test_status = 'F' or test_status = 'LC')

```

There were three indexes on the MIC_REQUESTED_TESTS table. Only one of these indexes had the two columns WORKSHEET_CODE and TEST_STATUS mentioned in the WHERE clause, and those columns were the leading segments of the index.

```
MRT_WORKSHEET_IDX          with column WORKSHEET_CODE
                           and column TEST_STATUS
                           and column LA_NUMBER
                           and column LOCATION_NUMBER
                           and column DATE_WORKSHEET_GENERATED
```

The following is a portion of the Rdb7 query strategy seen by enabling the strategy output first using the SQL statement SET FLAGS 'STRATEGY'. The Rdb7 strategy shows that dynamic optimization was chosen (by the presence of the keyword Leaf) and that retrieval was to be done using only the first segment of the index MRT_WORKSHEET_IDX (by the notation [1:1]).

```
Leaf#01 BgrOnly MIC_REQUESTED_TESTS Card=310030
BgrNdx1 MRT_WORKSHEET_IDX [1:1] Bool Fan=58
```

One would expect instead to see an indexed retrieval using the first two segments of the index and using a range list of index keys. The expected notation should be [(2:2)2] rather than [1:1], leading to more precise data retrieval and better query performance.

The problem was dependent on the order in which the candidate indexes were created. The good strategy could be obtained by first dropping all but the desired index and then re-creating the dropped indexes.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.27 Outer Join Query Bugchecks When Its Equi-join Predicate is Transitive With a Subselect Query

Bug 1295035

An outer join query would bugcheck when its equi-join predicate was transitive with a subselect query, as in the following example:

```
att 'f personnel';
select t0.employee_id
      ,( select count(*) from employees t3
         where t3.employee_id = t0.employee_id)
from
  employees t0,
  departments t1 left outer join salary_history t2 on
    t1.manager_id = t2.employee_id
where
  t0.employee_id = t1.manager_id ;
```

The key parts of this query which contributed to the situation leading to the error are these:

1. The query joins a table with the output of a left outer join, using the equi-join predicates between the table column and one of the left outer join columns
2. The table column is also part of an equi-join belonging to a subselect query

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.28 Bugcheck at PSII2SPLITNODE + 3A8

Bug 1205771

During insertions of duplicates into a ranked index duplicate, a problem in the way residual room in the index node is determined may cause a bugcheck at PSII2SPLITNODE + 3A8.

This problem occurs only rarely and only with ranked indexes with the following conditions:

- The insertion of dbkeys is occurring in increasing value order, as in the case of RMU/LOAD
- An index duplicate entry has a large enough number of duplicate dbkeys to cause the creation of an overflow node for that entry on the next dbkey insertion
- The next dbkey being inserted into the duplicate entry has to be either 65536 data pages greater than the first dbkey within that entry or have a different physical area number.

A workaround for this problem is to use a normal sorted index instead of a ranked index. Alternatively, the ranked index may be dropped during the data loading phase and recreated after the load has completed.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.29 Processes Vanished When Using Ranked Indexes

Bug 1427371

Oracle Rdb7 processes that utilized a sorted ranked index would sometimes disappear. Examination of the system error log would reveal a non-fatal bugcheck had occurred. If the SYSGEN parameter BUGCHECKFATAL was set to true and a system bugcheck occurred, the system bugcheck would reveal that the process had encountered a "NATIVE_TO_TRANSLATED" error.

This error would occur whenever an exception was raised in the sorted ranked index subsystem. For example, a deadlock error could be encountered. The exception handlers were not properly defined in the sorted ranked index code, and the exception handlers would attempt to call random addresses for exception handling. This, in turn, would often cause a VMS exception to occur, and ultimately lead to process deletion.

To workaround this problem, redefine the indexes to be regular sorted indexes instead of sorted ranked indexes.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.30 Query With COALESCE Function Returns Wrong Results

Bug 1415286

The following query with the COALESCE function returns wrong results.

```
select count(*) from salary_history s1
  where s1.salary_start >
        (select COALESCE (MAX(s2.salary_start), date vms '1-jan-1970')
         from salary_history s2 where
          s2.SALARY_AMOUNT > 100000 and
          s2.employee_id = s1.employee_id) ;
```

This problem is caused by the enhancement to improve the performance of the COALESCE function plus the fix for Bug 498674 in Oracle Rdb7 Release 7.0.1.

This problem occurs when the main query has a predicate which compares a column of a table with a subquery where COALESCE function is applied on an aggregate function (e.g. MIN, MAX, SUM, AVG) of another joined column.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.31 Query with DISTINCT, GROUP BY and ORDER BY Returns Wrong Results

Bug 1412456

The following query returns wrong results if a new row of duplicate maximum_salary is added to jobs:

```
att 'f personnel';
set flags 'strategy';

SELECT DISTINCT SUM(maximum_salary) FROM jobs GROUP BY maximum_salary
ORDER BY maximum_salary
LIMIT TO 3 ROWS;
Firstn Reduce Sort Aggregate Sort Get
Retrieval sequentially of relation JOBS
15000.00
17000.00
17500.00
3 rows selected
```

Now the first row should be 30000 instead of 15000:

```
insert into jobs (job_code, maximum_salary) value ('TEMP',15000);

SELECT DISTINCT SUM(maximum_salary) FROM jobs GROUP BY maximum_salary
ORDER BY maximum_salary
LIMIT TO 3 ROWS;
Firstn Reduce Sort Aggregate Sort Get
Retrieval sequentially of relation JOBS
15000.00
17000.00
17500.00
3 rows selected
```

This problem occurs when the key order of the ORDER BY clause in the query is the same as the GROUP BY clause, but different from the DISTINCT clause. Oracle Rdb7 optimizes away the sort node for ORDER BY key, since it is the same as GROUP BY, but never realizes that this sort order is required due to the different key in the DISTINCT clause.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.32 Query With CAST Function on CURRENT_DATE Returns Wrong Results

Bug 1459038

The following query with CAST function on CURRENT_DATE should return 5 rows:

```

select spcode from subs
  where sl='GL298' and
        dt_complete < cast (current_date as date vms) ;
Tables:
  0 = SUBINADC
  1 = SUBINADC2
Reduce Sort
Merge of 2 entries
Merge block entry 1
  Conjunct: ((0.SL = 'GL298') AND (0.DT_COMPLETE < CAST (<datatype:35> AS
        DATE VMS)))
  Get      Retrieval sequentially of relation SUBINADC
Merge block entry 2
  Conjunct: ((1.SL = 'GL298') AND (1.DT_COMPLETE < CAST (<datatype:35> AS
        DATE VMS)))
  Get      Retrieval sequentially of relation SUBINADC2
0 row selected.

```

This problem occurs when the query uses the CAST function CURRENT_DATE, which is also a function. This problem is caused by the fix made for Bug 1369801.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.33 Aggregate Query With Partitioned Index Bugchecks When SQL92 Dialect is Set

Bug 1312430

The following MAX function query with partitioned index bugchecks when SQL92 dialect is explicitly set:

```

set dialect 'sql92'
select MAX(BUSINESS_CALENDAR)
  from BUS_PRCSS_EVENT BPE
  where BPE.BPPF01FK_SYSTEM_CD = 'CPM' AND
        BPE.BPPF01FK_PROC_CD   = 'CROLL' AND
        BPE.PROCESS_STATUS_COD = 'ENDED';

```

where the table BUS_PRCSS_EVENT has a partitioned index defined as follows:

```

create index bus_prCSS_event_s_09
  on bus_prCSS_event
  (BPPF01FK_SYSTEM_CD
  ,PROCESS_STATUS_COD
  ,BPPF01FK_PROC_CD
  ,BUSINESS_CALENDAR desc
  ,BPPF01FK_CYCLE_CD desc
  ,CREATE_TS desc
  ,BPPF01FK_PROFIL_CD desc
  )
type is sorted
  node size 488
  percent fill 100
enable compression (minimum run length 2)
store using (bppf01fk_system_cd)
  in execution_control_indx_001 with limit of ('CBK ')
  in execution_control_indx_002 with limit of ('CPB ')
  in execution_control_indx_003 with limit of ('CPD ')
  in execution_control_indx_004 with limit of ('CPM ')
  in execution_control_indx_005 with limit of ('CST ')
  in execution_control_indx_006 with limit of ('CTM ')
  otherwise in execution_control_indx_007
;

```


The key parts of this query which contributed to the situation leading to the error are these:

1. The query has an aggregate function such as MAX
2. The index on the table is partitioned
3. SQL command "set dialect 'SQL92'" is issued before the query

As a workaround to this problem, turn off the SQL92 feature.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.1.34 Complex Nested View Query Returns Wrong Results When Match Strategy is Used

Bug 1353742

The following complex nested view query returns wrong results when the match strategy is used:

```
select company_no, account_no, department_no, department_name
from view_inquiry
where
  account_type_cd like '%' and
  (account_no BETWEEN 4110 and 4115) and
  company_no=1
;
```

Tables:

```
0 = GL_STARTING_BALANCE
1 = GL_STARTING_BALANCE
2 = GL_ACCOUNT
3 = GL_ACCOUNT_GROUP
4 = GL_ACCOUNT_TYPE
5 = DEPARTMENT
```

Cross block of 2 entries

Cross block entry 1

Match

Outer loop

Cross block of 2 entries

Cross block entry 1

Match

Outer loop

====>

(Missing sort on COMPANY_NO ?)

Conjunct: (4.ACCOUNT_TYPE_CD LIKE '%')

Get Retrieval sequentially of relation 4:GL_ACCOUNT_TYPE

Inner loop (zig-zag)

Get Retrieval by index of relation 2:GL_ACCOUNT

Index name GL_ACCOUNT_PMY [1:1] Bool

Key: (2.ACCOUNT_NO >= 4110) AND (2.ACCOUNT_NO <= 4115)

Bool: (2.COMPANY_NO = 1)

Cross block entry 2

Conjunct: (3.ACCOUNT_TYPE_CD LIKE '%')

Conjunct: ((3.COMPANY_NO = 4.COMPANY_NO) AND

(3.ACCOUNT_TYPE_CD = 4.ACCOUNT_TYPE_CD))

Get Retrieval by index of relation 3:GL_ACCOUNT_GROUP

Index name GL_ACCOUNT_GROUP_PMY [2:2] Bool Direct lookup

Key: (2.ACCOUNT_GROUP_NO = 3.ACCOUNT_GROUP_NO) AND

(2.COMPANY_NO = 3.COMPANY_NO)

Bool: (3.COMPANY_NO = 1)

Inner loop (zig-zag)

Get Retrieval by index of relation 5:DEPARTMENT

Index name DEPARTMENT_PMY [1:1]

Key: (5.COMPANY_NO = 1)

Cross block entry 2

Reduce Sort

```

Conjunct: ((2.COMPANY_NO = <based on field>0) AND
           (2.ACCOUNT_NO = <based on field>0) AND
           (5.COMPANY_NO = <based on field>0) AND
           (5.DEPARTMENT_NO = <based on field>0))
Merge of 2 entries
Merge block entry 1
Conjunct: (0.ACCOUNT_NO <= 4115)
Conjunct: ((0.ACCOUNT_NO >= 4110) AND (0.COMPANY_NO = 1))
Get      Retrieval by index of relation 0:GL_STARTING_BALANCE
Index name  GL_STARTING_BALANCE_PRMV [3:3] Bool      Direct lookup
Key: (5.DEPARTMENT_NO = <based on field>0) AND
     (2.ACCOUNT_NO = <based on field>0) AND
     (2.COMPANY_NO = <based on field>0)
Bool: (0.ACCOUNT_NO >= 4110) AND (0.ACCOUNT_NO <= 4115) AND
     (0.COMPANY_NO = 1)
Merge block entry 2
Conjunct: (1.ACCOUNT_NO <= 4115)
Conjunct: ((1.ACCOUNT_NO >= 4110) AND (1.COMPANY_NO = 1))
Get      Retrieval by index of relation 1:GL_STARTING_BALANCE
Index name  GL_STARTING_BALANCE_PRMV [3:3] Bool      Direct lookup
Key: (5.DEPARTMENT_NO = <based on field>0) AND
     (2.ACCOUNT_NO = <based on field>0) AND
     (2.COMPANY_NO = <based on field>0)
Bool: (1.ACCOUNT_NO >= 4110) AND (1.ACCOUNT_NO <= 4115) AND
     (1.COMPANY_NO = 1)
0 rows selected

```

Where the following tables and views are defined:

```

CREATE TABLE gl_account (
  company_no tinyint,
  account_no integer,
  account_group_no integer
);

CREATE TABLE gl_account_group (
  company_no tinyint,
  account_group_no integer,
  account_group_desc char (25),
  account_type_cd char (1)
);

CREATE TABLE gl_account_type (
  company_no tinyint,
  account_type_cd char (1),
  account_type_desc char (25)
);

CREATE TABLE department (
  company_no tinyint,
  department_no integer,
  department_name char (25)
);

CREATE TABLE gl_trx_hist (
  company_no tinyint,
  account_no integer,
  department_no integer
);

CREATE TABLE gl_starting_balance (
  company_no tinyint,
  account_no integer,
  department_no integer,
  starting_balance integer
);

```

```

create unique index DEPARTMENT_PRMV
  on DEPARTMENT (COMPANY_NO, DEPARTMENT_NO) ;

create unique index GL_ACCOUNT_GROUP_PRMV
  on GL_ACCOUNT_GROUP (ACCOUNT_GROUP_NO, COMPANY_NO);

create unique index GL_ACCOUNT_PRMV
  on GL_ACCOUNT (ACCOUNT_NO, COMPANY_NO);

create unique index GL_STARTING_BALANCE_PRMV
  on GL_STARTING_BALANCE (ACCOUNT_NO, DEPARTMENT_NO, COMPANY_NO);

create view vw_account (
  company_no,
  account_no,
  account_group_no,
  account_type_cd
) AS
SELECT  c1.company_no, c1.account_no, c1.account_group_no,
        c2.account_type_cd
FROM    gl_account c1,
        gl_account_group c2,
        gl_account_type c3
WHERE   c1.company_no = c2.company_no AND
        c2.company_no = c3.company_no AND
        c1.account_group_no = c2.account_group_no AND
        c2.account_type_cd = c3.account_type_cd ;

create view vw_balance
  (company_no,
   account_no,
   department_no,
   starting_balance) AS
SELECT
  c2.company_no, c2.account_no, c2.department_no, c2.starting_balance
  FROM gl_starting_balance c2
UNION
SELECT
  c2.company_no, c2.account_no, c2.department_no, c2.starting_balance
  FROM gl_starting_balance c2;

CREATE view vw_inquiry (
  company_no,
  account_no,
  department_no,
  account_type_cd,
  account_group_no,
  department_name )
AS
SELECT
  glb.company_no, glb.account_no, glb.department_no,
  gla.account_type_cd, gla.account_group_no,
  dpt.department_name
  from vw_balance glb,
       vw_account gla,
       department dpt
  where
    gla.company_no = glb.company_no and
    gla.account_no = glb.account_no and
    dpt.company_no = glb.company_no and
    dpt.department_no = glb.department_no ;

commit work;

insert into gl_account_type (company_no,account_type_cd) value (1, '0');
insert into gl_account_type (company_no,account_type_cd) value (2, '0');
insert into gl_account_type (company_no,account_type_cd) value (1, 'R');
insert into gl_account_type (company_no,account_type_cd) value (2, 'R');

```

```

insert into gl_account (company_no,account_no,account_group_no)
value (1, 4115, 4100);

insert into gl_account_group (company_no,account_group_no,account_type_cd)
value (1, 4100, 'r');

insert into department (company_no, department_no) value (1, 0);

insert into gl_starting_balance (company_no, account_no, department_no)
value (1, 4115, 0);

```

This problem occurs when the query has selection predicates on a view of 3-way equality join, where a table is joined to a view (with union legs), and another view (with 3-way equality join on 3 tables).

One of the selection predicates (i.e. account_type_cd like '%') is then transitively solved on one table of the 3-tables view as a match strategy without sorting on the match key (i.e. COMPANY_CO).

The workaround is to define the logical RDMS\$SET_FLAGS, or use the SQL SET FLAGS statement with the value NOTTRANSITIVITY or define the logical name RDMS\$TRANSITIVITY to FALSE.

When this flag is defined, the transitivity feature is disabled and the query uses the cross strategy, instead of match, and thus produces the correct results.

This problem was introduced by a fix made for Bug 926540 in Oracle Rdb7 Release 7.0.3.1.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.2 SQL Errors Fixed

2.2.1 Unexpected Column Name Present in SQLDA Structure

In prior versions of Oracle Rdb, the SQL interface would eliminate duplicate aggregate functions within a query select list. This optimization was designed to reduce the calculation overhead of the query. For example, the SUM expressions are identical in the following query.

```

SQL> select SUM (salary_amount) as X,
cont>        SUM (salary_amount) as Y
cont> from salary_history;

```

	X	X
	19339617.00	19339617.00

```

1 row selected
SQL>

```

Although the displayed results are correct, the column name provided by the AS clause has been lost. In this interactive SQL example, the column heading is simply incorrect. However, this incorrect column name is also passed in the SQLDA structure and may cause applications which use the Dynamic SQL interface to fail.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.2.2 Unexpected Errors from Date/Time Subtraction

Bug 1164958

When date/time values are subtracted using the ANSI and ISO SQL Standard syntax, the result is an INTERVAL type; either a year-month interval, or day-time interval. These interval types are mutually exclusive and so programmers must specify the type of interval result using an interval qualifier on the subtraction.

```
SQL> select (current_date - birthday) YEAR TO MONTH
cont> from employees where employee_id='00276';
```

```
42-09
1 row selected
```

However, when the dialect is set to ORACLE LEVEL1, Rdb will supply an implicit interval qualifier of DAY and then extract this value as an integer (using the EXTRACT function). This provides a result that is compatible with Oracle RDBMS, which only supports date subtraction results as the number of days.

However, as these examples show, this syntax was not always handled correctly in prior releases of Oracle Rdb.

```
SQL> set dialect 'oracle level1';
SQL> set flags 'trace';
SQL>
SQL> begin
cont> declare :x date vms default date vms'1-JUN-1999';
cont> declare :y date vms default date vms'9-JUN-1999';
cont> declare :z integer;
cont> set :z = :x - :y;
cont> trace :z;
cont> end;
%SQL-F-INVINTQUAL, Invalid interval qualifier
SQL>
SQL> begin
cont> declare :x, :y date = current_timestamp;
cont> trace
cont> case :y - :x
cont>   when 1 then 'yesterday'
cont>   when -1 then 'tomorrow'
cont>   when 0 then 'today'
cont>   else '****'
cont> end;
cont> end;
%SQL-I-NUMCMPTXT, Numeric column will be compared with string literal as text
%SQL-I-NUMCMPTXT, Numeric column will be compared with string literal as text
%SQL-I-NUMCMPTXT, Numeric column will be compared with string literal as text
%SQL-I-BUGCHKDMP, generating bugcheck dump file DISK1:[TEST]SQLBUGCHK.DMP;
%SQL-F-BUGCHK, There has been a fatal error. Please contact your Oracle support
representative. SQL$BLRXPR - 25
SQL>
SQL> begin
cont> declare :x, :y date = current_timestamp;
cont> trace DECODE (:y - :x,
cont>   1, 'yesterday',
cont>  -1, 'tomorrow',
cont>   0, 'today', '****');
cont> end;
%SQL-I-BUGCHKDMP, generating bugcheck dump file DISK1:[TEST]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=0000000000000065, PC=00000000002D4EF0, PS=0000001B
```

These problems have been corrected in Oracle Rdb7 Release 7.0.6.

2.2.3 SQL Functions Can Now Be Called From Triggers

In prior versions of Oracle Rdb7, only external functions could be called from a trigger condition (WHEN clause), or trigger action (INSERT, UPDATE or DELETE). The error that was generated is shown in this example.

```
SQL> create module M_MODULE
cont>   language SQL
cont>
cont>   function M_K (in :a int) returns int;
cont>   return case
cont>       when :a <= 0 then 1
cont>       when :a is null then 1
cont>       else :a
cont>   end;
cont> end module;
SQL>
SQL> create table M_TABLE1 (a integer, b integer);
SQL> create table M_TABLE2 (a integer, b integer);
SQL> create trigger T_A
cont>   after insert on M_TABLE2
cont>   (update M_TABLE1
cont>   set b = M_K (M_TABLE2.a)
cont>   where a = M_TABLE2.a)
cont>   for each row;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-RTN_FAIL, routine "M_K" failed to compile or execute successfully
-RDMS-E-NOTRIGRTN, a stored routine may not be called from a trigger
```

With this release of Rdb7, this restriction has been relaxed so that SQL stored functions can be executed from triggers with the following restrictions:

- the SQL function must not execute an INSERT, DELETE, or UPDATE statement
- the SQL function may not use a CALL statement, or activate another stored function in a value expression

This problem has been corrected in Oracle Rdb7 Release 7.0.6. SQL functions which use other procedural statements or call external routines may be called from a trigger.

Using existing SQL functions

SQL functions created by prior releases which conform to these restrictions may still be rejected by CREATE TRIGGER. This is because the correct execute state has not been recorded for this routine. This can be corrected by using DROP and CREATE to create a new version.

An alternate method is to use the SET FLAGS 'VALIDATE_ROUTINE' option as shown in this example. Once the validation has been performed, a COMMIT will store the state in the RDB\$ROUTINES system table for future use.

```

SQL> set transaction read write;
SQL>
SQL> create table M_TABLE1 (a integer, b integer);
SQL> create table M_TABLE2 (a integer, b integer);
SQL>
SQL> -- attempt to use SQL function fails
SQL> create trigger T_A
cont>     after insert on M_TABLE2
cont>     (update M_TABLE1
cont>       set b = ADD_ONE (M_TABLE2.a)
cont>       where a = M_TABLE2.a)
cont>     for each row;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-RTN_FAIL, routine "ADD_ONE" failed to compile or execute successfully
-RDMS-E-NOTRIGRTN, this stored routine may not be called from a trigger
SQL>
SQL> set flags 'VALIDATE_ROUTINE';
SQL> -- use NOEXECUTE so the routine is just compiled, not executed
SQL> set noexecute;
SQL> -- validate the routine to set the correct routine state
SQL> select ADD_ONE (1) from rdb$database;
0 rows selected
SQL>
SQL> set execute;
SQL>
SQL> -- now the routine can be successfully used with the trigger definition
SQL> create trigger T_A
cont>     after insert on M_TABLE2
cont>     (update M_TABLE1
cont>       set b = ADD_ONE (M_TABLE2.a)
cont>       where a = M_TABLE2.a)
cont>     for each row;
SQL>
SQL> commit;

```

2.2.4 Oracle LEVEL1 Dialect Not Displayed by SHOW CONNECTIONS

Bug 734982

In prior releases of Oracle Rdb7, the SHOW CONNECTIONS statement would display the dialect as SQL92 even though a SET DIALECT 'ORACLE LEVEL1' was used for the session.

While this was not incorrect - Oracle LEVEL1 dialect is a superset of the SQL92 - it did not make it clear that a SET DIALECT included Oracle LEVEL1 semantics. Therefore, for this release of Rdb, the SHOW CONNECTIONS statement will also include the Oracle LEVEL information.

The following example shows the new output.

```

SQL> set dialect 'sql92';
SQL> show connect RDB$DEFAULT_CONNECTION
Connection: RDB$DEFAULT_CONNECTION
.
.
.
Dialect: SQL92
.
.
.
SQL> set dialect 'ORACLE LEVEL1';
SQL> show connect RDB$DEFAULT_CONNECTION
Connection: RDB$DEFAULT_CONNECTION
.
.
.
Dialect: SQL92 (ORACLE LEVEL1)
.
.
.

```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.2.5 GET DIAGNOSTICS Option TRANSACTION_CHANGE_ALLOWED Errors

The GET DIAGNOSTICS option TRANSACTION_CHANGE_ALLOWED did not always return the correct values. For instance, if GET DIAGNOSTICS were used in a FOR cursor loop, it could return 1 (TRUE), even though using COMMIT, ROLLBACK or SET TRANSACTION are not permitted at this location.

Most of these problems have been corrected in Oracle Rdb7 Release 7.0.6.

In addition the following restrictions continue to apply to this function:

- Changing the transaction state in a compound statement (or stored procedure) is not permitted when a transaction spans more than one database. The COMMIT or ROLLBACK must be performed from the client application (not the database server) and be coordinated across the databases. However, if a transaction is not currently active, then the procedure does not know that multiple aliases are involved. Therefore, the returned value of TRANSACTION_CHANGE_ALLOWED will return 1 (TRUE) even though an attempt will generate an error.

The workaround for this problem is to avoid using the result of TRANSACTION_CHANGE_ALLOWED unless a transaction is active (see the GET DIAGNOSTICS option TRANSACTION_ACTIVE).

- Neither SET TRANSACTION, COMMIT nor ROLLBACK are permitted within an ATOMIC compound statement. However, SQL currently optimizes out the ATOMIC attribute when only a GET DIAGNOSTICS is present. Therefore, the returned value of TRANSACTION_CHANGE_ALLOWED will return 1 (TRUE) even though an attempt to change the transaction state would generate an error.

The workaround for this problem is to call a stored function which processes the GET DIAGNOSTICS. In this case, the ATOMIC attribute is recognized.

2.2.6 Temporary File Cleanup During Precompiled SQL

Bug 478898

The SQL precompiler generates some temporary files (like .mli, .mob) that are now cleaned up after compilation unless the `SQL$KEEP_PREP_FILES` logical has been defined.

The following example shows the file .mob left around on OPENVMS/Alpha when the sql precompilation fails:

```
$ sqlpre/cc scc_try_mli_bad_sql.sc
exec sql select count(*) :howmany from rdb$relations
          1
%SQL-F-SYM_EXP, (1) One of the following symbols was expected:
%SQL-F-EXPSYMLIS,      || * / + - AS name , INTO FROM WHERE
%SQL-I-ERRSYMREP, Error symbol replaced by ,
$ dir/size scc_try_mli_bad_sql.*

SCC_TRY_MLI_BAD_SQL.C;1 21
SCC_TRY_MLI_BAD_SQL.MOB;1 1
SCC_TRY_MLI_BAD_SQL.SC;1 16

Total of 3 files, 38 blocks.
```

As a workaround to this problem, delete the .mob file after the unsuccessful precompilation.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.2.7 SET TRANSACTION Looses RESERVING Clause

Bug 1364647

In prior versions of Oracle Rdb, the `SET TRANSACTION` and `DECLARE TRANSACTION` statements did not correctly process the tables in the `RESERVING` clause when more than one alias was used with the `ON ... USING` clause.

- If a default alias was used and the tables listed in the `RESERVING` clause were not qualified by an alias, they could be lost.

For instance, the following example attempts to reserve `EMPLOYEES` in alias `T` (the `MF_PERSONNEL` database). Unfortunately, SQL assumes the default alias `RDB$DBHANDLE` for the table and quietly omits the reserving clause (as shown by the `TRANSACTION` debug trace in the example).

```

SQL> attach 'filename SCRATCH';
SQL> attach 'alias t filename MF_PERSONNEL';
SQL>
SQL> set flags 'noprefix,transaction';
SQL>
SQL> set transaction
cont>     on rdb$dbhandle using (read only)
cont>     and
cont>     on t using (read write
cont>         reserving employees for exclusive write);
  Compile transaction on db: X00000001
~T Transaction Parameter Block: (len=2)
TPB$K_VERSION = 1
TPB$K_READ (read only)
  Start_transaction on db: X00000001, db count=2
  Compile transaction on db: X00000002
~T Transaction Parameter Block: (len=2)
TPB$K_VERSION = 1
TPB$K_WRITE (read write)
  Start_transaction on db: X00000002, db count=2

```

- **Related to this problem, is the case where a table is qualified with an alias other than the one for the containing ON ... USING clause. Again, SQL quietly ignores the table in the reserving list. Only the correctly specified table is used.**

```

SQL> set transaction
cont>     on rdb$dbhandle using (read only)
cont>     and
cont>     on t using (read write
cont>         reserving t.employees,
cont>         rdb$dbhandle.sample for exclusive write);
  Compile transaction on db: X00000001
~T Transaction Parameter Block: (len=2)
TPB$K_VERSION = 1
TPB$K_READ (read only)
  Start_transaction on db: X00000001, db count=2
  Compile transaction on db: X00000002
~T Transaction Parameter Block: (len=14)
TPB$K_VERSION = 1
TPB$K_WRITE (read write)
TPB$K_LOCK_WRITE (reserving) "EMPLOYEES" TPB$K_EXCLUSIVE
  Start_transaction on db: X00000002, db count=2

```

The workaround for these problems is to include the alias on all tables in the reserving list and to ensure that it is the alias specified in the enclosing ON ... USING clause.

These problems have been corrected in Oracle Rdb7 Release 7.0.6. Now when the alias is omitted, SQL defaults to the enclosing ON ... USING alias and correctly processes the RESERVING list. If an incorrect alias is specified, then an exception is raised, either DBHANDUNK, or CONFXTXNALIAS.

```

SQL> set transaction
cont>     on rdb$dbhandle using (read only)
cont>     and
cont>     on t using (read write
cont>         reserving tt.employees for exclusive write);
%SQL-F-DBHANDUNK, TT is not the alias of a known database

```

```

SQL> set transaction
cont>     on rdb$dbhandle using (read only)
cont>     and
cont>     on t using (read write
cont>         reserving rdb$dbhandle.sample for exclusive write);
%SQL-F-CONFTEXTN_ALIAS, Expected alias "T" (found "RDB$DBHANDLE") for table in
reserving clause

```

2.2.8 Unexpected Informational Message Issued for Some Statements

Bug 1067290

When the dialect was set to SQL92 or ORACLE LEVEL1, it was possible in prior versions of Oracle Rdb to see an erroneous warning message issued for SHOW, SET, HELP and similar statements. This is demonstrated by the following example.

```

SQL> set dialect 'sql92';
SQL> attach 'file MF_PERSONNEL';
SQL>
SQL> update employees
cont>     set state = NULL
cont>     where employee_id < '00170';
6 rows updated
SQL>
SQL> select count(*), count(state) from employees;

           100           94
1 row selected
%RDB-I-ELIM_NULL, null value eliminated in set function
SQL>
SQL> show table (col) candidates
Information for table CANDIDATES

Columns for table CANDIDATES:
Column Name                Data Type                Domain
-----
LAST_NAME                   CHAR(14)                  LAST_NAME_DOM
  Not Null constraint CANDIDATES_LAST_NAME_NOT_NULL
FIRST_NAME                   CHAR(10)                  FIRST_NAME_DOM
MIDDLE_INITIAL               CHAR(1)                   MIDDLE_INITIAL_DOM
CANDIDATE_STATUS             VARCHAR(255)              CANDIDATE_STATUS_DOM

%RDB-I-ELIM_NULL, null value eliminated in set function
SQL>

```

This message is incorrectly being retained from a previous statement execution and should not be displayed. This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.2.9 Oracle RDBMS Style Join Syntax Does Not Support ALL Predicate

Bug 1482449

In prior releases of Oracle Rdb7, SQL would generate a bugcheck if the ALL predicate was used in the same query as the Oracle RDBMS style outer join syntax. This is shown in the example below.

```

select emps.employee_id, emps.last_name
  from employees emps, degrees degs, colleges colls
  where emps.employee_id(+) = degs.employee_id
     and degs.college_code <> all (select college_code from colleges colls)
;

```

When run, the following error is displayed:

```
%SQL-I-BUGCHKDMP, generating bugcheck dump file DISK1:[BZINN]SQLBUGCHK.DMP;
%SQL-F-BUGCHK, There has been a fatal error. Please contact your Oracle
support representative. SQL$SEMRSE - 71
```

In the bugcheck, an exception such as the following will be found:

```
***** Exception at 001E7340 : SQL$$WALK_SEQUENCE_VALUE + 00001170
%SQL-F-BUGCHK, There has been a fatal error. Please contact your Oracle
support representative. SQL$SEMRSE - 71
```

The workaround for this example is to replace the '<> ALL' with 'NOT IN'.

The following example shows this work around:

```
select emps.employee_id, emps.last_name
      from employees emps, degrees degs, colleges colls
      where emps.employee_id(+) = degs.employee_id
            and degs.college_code not in (select college_code from colleges colls)
      ;
```

This problem has been corrected in Oracle Rdb7 Release 7.0.6. SQL now supports the ALL operator in conjunction with the Oracle RDBMS style outer join syntax.

2.2.10 DROP STORAGE AREA ... CASCADE May Fail With ACCVIO Error

Bug 1408875

In prior releases of Oracle Rdb7, the ALTER DATABASE ... DROP STORAGE AREA ... CASCADE command may fail when the RDMS\$DEBUG_FLAGS logical includes "As", or the RDMS\$SET_FLAGS logical includes "STOMAP_STATS". These values enable tracing of storage map statistics in numerous SQL DDL statements.

```
$ define rdms$set_flags "stomap_stats,index_stats"
$ sql$
SQL> alter database file testdb70
cont> drop storage area testdb70_mixed_51 cascade;
~As: Drop Storage Area "TESTDB70_MIXED_51" Cascade
~As: ...area referenced by map: "TAB5_MAP"
~As: ...area referenced by index: "TAB5_HNDX"
%RDB-F-SYS_REQUEST, error from system services request
-COSI-F-FILACCERR, error formatting file
-COSI-F-ACCVIO, access violation
```

The problem results from an error formatting the trace message for indices. The workaround is to deassign these logical names so that no trace messages are generated.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.2.11 Exception in Nested Routines Did Not Close Index Scans

Bug 1058528

In prior releases of Oracle Rdb7, exceptions occurring in a nested stored procedure or function could leave open index scans. This problem resulted in bugchecks during the ROLLBACK statement. The bugcheck would have a signature similar to the following:

```
***** Exception at 03435E80 : KOD$ROLLBACK + 000001D8
%COSI-F-BUGCHECK, internal consistency failure
```

To cause this problem, an exception must be raised while a query is scanning an index. A nested stored routine is one invoked by the CALL statement or a function expression within a compound statement (BEGIN END block).

The exception may occur in the routine body itself, a trigger action, or in a nested stored routine. There is no workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

Note

The routine KOD\$ROLLBACK implements the rollback functionality. It contains a sanity check to ensure correct operation of the Rdb server. While every effort has been made to correct this reported problem, it is possible that some unrelated problem may produce a similar bugcheck signature. Please report all future occurrences to Oracle Support.

2.3 Oracle RMU Errors Fixed

2.3.1 /OPEN_MODE and /CLOSE_WAIT for RMU/RESTORE and /COPY

In Oracle Rdb7 Release 7.0.6, the existing /OPEN_MODE qualifier used with the RMU/RESTORE and RMU/COPY commands will also set the database close mode. In addition, a new optional "/CLOSE_WAIT=n" qualifier has been added to allow the user to specify a wait time of "n" minutes before automatically closing the database.

Starting with this release, the database open mode setting of "AUTOMATIC" or "MANUAL" specified with the /OPEN_MODE qualifier of the RMU/RESTORE command will also set the database close mode as with the SQL ALTER DATABASE "OPEN IS" clause. In addition, since the SQL ALTER DATABASE "OPEN IS" clause has a "(WAIT n MINUTES FOR CLOSE)" option, an optional /CLOSE_WAIT=n qualifier, where n is the number of minutes to wait before automatically closing the database, has been added to the RMU/RESTORE and RMU/COPY commands. The /CLOSE_WAIT qualifier can only be used if /OPEN_MODE=AUTOMATIC is also specified and a value must be specified with this qualifier.

Therefore, if /OPEN_MODE=MANUAL is specified, the open and close modes are set to MANUAL. If /OPEN_MODE=AUTOMATIC is specified, the open and close modes are set to AUTOMATIC. If /OPEN_MODE=AUTOMATIC/CLOSE_WAIT=n is specified, the open mode is set to AUTOMATIC and the close mode is set to TIMED AUTOMATIC.

The following example uses the new option.

```
RMU/RESTORE/OPEN_MODE=MANUAL MF_PERSONNEL.RBF
RMU/RESTORE/OPEN_MODE=AUTOMATIC MF_PERSONNEL.RBF
RMU/RESTORE/OPEN_MODE=AUTOMATIC/CLOSE_WAIT=5 MF_PERSONNEL.RBF
RMU/COPY/OPEN_MODE=AUTOMATIC/CLOSE_WAIT=5 MF_PERSONNEL /DIRECTORY=DEV:[DIR]
```

2.3.2 RMU Online Backup Consumes Excessive CPU Resources

In some large, complex databases with a large number of Row Caches defined or reserved, RMU online backup operations may consume excessive CPU time.

This situation has been improved. Several optimizations have been made to RMU/BACKUP to help reduce the CPU usage during online backups. These improvements will be most noticeable on databases with row caches defined or reserved.

2.3.3 SHOW STATS "Report" Does Not Include Logical Area "Graphs"

When you generate the "Statistics Report" from the RMU Show Statistic utility, via the on-screen menu "Option" using "B. Write Report (Numbers)", you get for the Summary IO Statistics the numeric screen in the Statistics.Rpt file.

When you generate a "Statistics Report" via the on-screen menu "Option" using "C. Write Report (Both)", you get for the summary IO Statistics both the numeric and graphical screens in the Statistics.Rpt file.

However, when you generate the Statistics report for the Logical Area Statistics, only the numeric screen is written to the Statistics.Rpt file.

This problem has been corrected in Oracle Rdb7 Release 7.0.6. When using the "C. Write Report (Both)" option, both numeric and graphical versions of all applicable screens are displayed.

2.3.4 RMU/UNLOAD/AFTER_JOURNAL Sort Avoidance

The RMU/UNLOAD/AFTER_JOURNAL performs a sort operation to eliminate duplicate record modifications for each transaction being extracted. This sort operation was initiated even when there were no records being extracted for the transaction. This sort is not needed and resulted in additional CPU resources being consumed in some cases.

This problem has been corrected in Oracle Rdb7 Release 7.0.6. The sort operation is avoided if possible.

2.3.5 RMU/UNLOAD/AFTER_JOURNAL Required All AIJ Backups to be Quiet-Point

In prior releases of Oracle Rdb, the RMU/UNLOAD/AFTER_JOURNAL command incorrectly required that all input AIJ backup files were "quiet-point" AIJ backups.

This problem has been corrected in Oracle Rdb7 Release 7.0.6. Only the *first* AIJ backup file supplied to the RMU/UNLOAD/AFTER_JOURNAL command must be a "quiet-point" AIJ backup; additional AIJ backup files do not need to be from "quiet-point" AIJ backups.

2.3.6 RMU /BACKUP /AFTER_JOURNAL Delay Reduced

Previously, the RMU /BACKUP /AFTER_JOURNAL command included a fixed delay of at least 60 seconds to allow database users to be notified of the database backup and after image journal switch. This delay has been reduced to 5 seconds to make the RMU /BACKUP /AFTER_JOURNAL command start faster.

2.3.7 RMU/LOAD Starts Read-Only Transaction

Previously, the RMU /LOAD utility always started a read-only transaction to read the database metadata. However, for databases with snapshots "enabled deferred", this transaction would be stalled until all other existing read/write transactions committed.

This problem has been corrected in Oracle Rdb7 Release 7.0.6. When a database is not set to snapshots "enabled immediate", the RMU /LOAD utility does not start a read-only transaction but instead uses a read/write transaction to read the database metadata.

2.3.8 RMU/UNLOAD/AFTER_JOURNAL Starts Read-Only Transaction

Previously, the RMU /UNLOAD /AFTER_JOURNAL utility always started a read-only transaction prior to reading the database metadata. However, for databases with snapshots "enabled deferred", this transaction would be stalled until all other existing read/write transactions committed.

This problem has been corrected in Oracle Rdb7 Release 7.0.6. When a database is not set to snapshots "enabled immediate", the RMU /UNLOAD /AFTER_JOURNAL utility does not start a read-only transaction, but instead uses a read/write transaction to read the database metadata.

2.3.9 RMU/SHOW STATISTIC /OUTPUT and /RESET Display Wrong Statistics Values

The RMU/SHOW STATISTIC utility displays incorrect statistics information when both the /OUTPUT and /RESET qualifiers are specified. The problem does not occur when only one or the other qualifier is specified.

For example, the default "Summary I/O Statistics" screen transactions will show a maximum value of 600 if the database is open on 6 nodes and an average value of 6.0. If the database is open on one node, max will be 100 and average will be 1.0. The same values come out for verb failures and data reads and writes. The same values come out for other statistics screens.

The display to the output file is correct. It is only the display to SYSS\$OUTPUT that is wrong and only if /OUTPUT and /RESET qualifiers are both specified.

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.10 RMU/EXTRACT Bugchecks During /ITEM=PROTECTION

Bug 1302921

RMU/EXTRACT does not correctly handle access control lists (ACL) with many entries (for example more than 80 access control entries). Attempts to extract these ACL's may result in an RMU/EXTRACT bugcheck.

```
%RMU-I-BUGCHECK, generating bugcheck dump file DISK:[DIR]RMUEXTBUGCHK.DMP;
```

A workaround is to locate the problem ACL and remove it prior to using RMU/EXTRACT/ITEM=PROTECTION.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.11 RMU/SHOW STATISTICS /UNTIL Hangs

Bug 824749

The RMU/SHOW STATISTICS utility would sometimes hang in "HIB" state when the /UNTIL qualifier was used.

There was a rare timing condition that would cause RMU/SHOW STATISTICS to neglect to exit when the /UNTIL time was reached.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.12 AIJ Quiet-Point Backup With Hot Standby and Row Cache Stops Checkpoint Advancement

AIJ quiet-point backups, when Hot Standby and Row Cache are enabled, or Hot Standby and the maximum node count set to "1", may result in the master database checkpoint not being able to be advanced. This results in the AIJ backup failing with the AIJRNBSY error because the AIJ required for backup contains the oldest active checkpoint, even when no application processes are active on the master database.

When Hot Standby is enabled, the oldest active checkpoint will be held by the "LCS" server process, which means the standby database did not advance its checkpoint. This problem can be identified by the master database being one journal in advance of the standby database. For example, the master database current AIJ sequence may be 25 and the standby database will be at 24.

This problem occurs for both the manual AIJ backup as well as the "ABS" server process.

The only workaround for this problem is to use no-quietpoint AIJ backups.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.13 Erroneous ABMBITERR Errors When Verifying Large Storage Areas

Bug 1329913

When verifying large uniform storage areas, it was possible for RMU/VERIFY to return erroneous ABMBITERR errors. For example:

```
%RMU-W-ABMBITERR, inconsistency between spam page 8475841 and bit 7777 in  
area bitmap in larea 2900 page 8102521
```

Although Verify reported an error, the database was actually structurally correct. This error would only occur when the area bit map (ABM) used to describe a table required more than one page. For example, if the SPAM interval for a storage area was 1089 pages, and an ABM page could hold 7776 entries, then it was possible for RMU/VERIFY to report undeserved ABMBITERR errors for pages beyond page 8468064 (1089 * 7776).

The problem was due to an "off by one" error in the Verify utility.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.14 Long Transaction Logfile and "Checkpoint Information" Screen Have Same Threshold

The RMU/Show Statistic utility "Long Transaction" logging facility and the "Checkpoint Information" screen share the same long transaction threshold. This makes it difficult to both log long-running transactions and view the "Checkpoint Information" screen at the same time.

There is no workaround to this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6. A new configuration variable, LONG_TX_SECONDS, has been added to identify the long-running transaction threshold, expressed in seconds. The default value is "1" second.

Also, the "Start stall logging" option of the Tools menu (shortcut "!") has been enhanced to prompt for the long-running transaction threshold, if any.

2.3.15 RMU/UNLOAD/AFTER_JOURNAL Checks Callback Routine Return Status

Previously, when using a "callback" routine with the RMU/UNLOAD/AFTER_JOURNAL command, the status returned from the routine was ignored. This prevented the callback routine from being able to control the extraction operation.

This problem has been corrected in Oracle Rdb7 Release 7.0.6. The RMU/UNLOAD/AFTER_JOURNAL command now checks the return status from a callback routine. The status returned from the callback routine is expected to be a valid format OpenVMS condition code. If the status is ODD (for example, the least significant bit is set), it indicates success. If the status is EVEN (for example, the least significant bit is clear), it indicates failure and the RMU/UNLOAD/AFTER_JOURNAL command will signal an exception and the extract operation will be terminated.

It is important that the callback routine always return a status value. A default value of SSS_NORMAL is suggested to indicate success.

2.3.16 RMU /RECOVER Fails When LogMiner Enabled

When the LogMiner(TM) feature is enabled and the database contains records longer than about 32767 bytes, it is possible for subsequent RMU /RECOVER (or RMU /RESTORE /RECOVER) commands to fail while reading the after-image journal file.

This problem was caused by the journaling of "pre-delete" content records for the LogMiner. If a "pre-delete" content record was larger than about 32767 bytes, the RMU /RECOVER code would be unable to correctly read and process the record.

This problem has been corrected in Oracle Rdb7 Release 7.0.6. Rdb now fragments a long "pre-delete" content record into parts smaller than 32500 bytes.

2.3.17 RMU /UNLOAD /AFTER_JOURNAL Output Records Larger Than 32767 Bytes

Previously, the RMU /UNLOAD /AFTER_JOURNAL was unable to output records larger than 32767 bytes to a file or device. This problem was due to a OpenVMS RMS limitation on record length.

This problem has been addressed in Oracle Rdb7 Release 7.0.6. Output records are now automatically broken into 32767 byte segments when writing to a file or device. There is no change in behavior when using a callback routine; the entire record is passed in one part.

2.3.18 RMU /UNLOAD /AFTER_JOURNAL Enhanced VARCHAR Support

When extracting tables in BINARY format with VARCHAR fields and creating .RRD (record definition) or .SQL (table definition) files, the output field definitions did not always match the binary data record. The VARCHAR data type includes a word (smallint) length field prior to the actual data bytes. This field was not represented in the .RRD (record definition) or .SQL (table definition) files.

This problem has been corrected in Oracle Rdb7 Release 7.0.6. A “virtual” field named “RDB\$LM_VARCHAR_LEN_n” (where “n” is a unique number within the record), is inserted prior to each VARCHAR field.

The following example .RRD file includes the new field:

```
DEFINE FIELD RDB$LM_ACTION DATATYPE IS TEXT SIZE IS 1.
DEFINE FIELD RDB$LM_RELATION_NAME DATATYPE IS TEXT SIZE IS 31.
DEFINE FIELD RDB$LM_RECORD_TYPE DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD RDB$LM_DATA_LEN DATATYPE IS SIGNED WORD.
DEFINE FIELD RDB$LM_NBV_LEN DATATYPE IS SIGNED WORD.
DEFINE FIELD RDB$LM_DBK DATATYPE IS SIGNED QUADWORD.
DEFINE FIELD RDB$LM_START_TAD DATATYPE IS DATE.
DEFINE FIELD RDB$LM_COMMIT_TAD DATATYPE IS DATE.
DEFINE FIELD RDB$LM_TSN DATATYPE IS SIGNED QUADWORD.
DEFINE FIELD RDB$LM_RECORD_VERSION DATATYPE IS SIGNED WORD.
DEFINE FIELD RDB$LM_VARCHAR_LEN_0 DATATYPE IS SIGNED WORD.
DEFINE FIELD V1 DATATYPE IS TEXT SIZE IS 20.
DEFINE FIELD RDB$LM_VARCHAR_LEN_1 DATATYPE IS SIGNED WORD.
DEFINE FIELD V2 DATATYPE IS TEXT SIZE IS 20.
DEFINE RECORD RDB_LM_C1.
    RDB$LM_ACTION.
    RDB$LM_RELATION_NAME.
    RDB$LM_RECORD_TYPE.
    RDB$LM_DATA_LEN.
    RDB$LM_NBV_LEN.
    RDB$LM_DBK.
    RDB$LM_START_TAD.
    RDB$LM_COMMIT_TAD.
    RDB$LM_TSN.
    RDB$LM_RECORD_VERSION.
    RDB$LM_VARCHAR_LEN_0.
    V1.
    RDB$LM_VARCHAR_LEN_1.
    V2.
END RDB_LM_C1 RECORD.
```

Note, however, that the actual VARCHAR data field contents are only valid up to the number of characters specified by the RDB\$LM_VARCHAR_LEN field. Any characters after that in the VARCHAR data field contain unpredictable content.

2.3.19 RMU OPTIMIZER_STATISTICS Leaves LOG File With Large Allocation

Bug 723307

In prior releases of Oracle Rdb7, various RMU OPTIMIZER_STATISTICS options (/SHOW, /INSERT and /DELETE) would create log files (/LOG) with a default extent of 512 blocks. The file was not explicitly closed on exit and the unused allocation would remain so that the output file would consume more disk space than was actually required.

The following example shows this behavior:

```
$ RMU/SHOW OPTIMIZER_STATISTICS SCRATCH/SYSTEM/LOG=QO.LOG  
$ DIRECTORY QO.LOG/SIZE=ALL
```

```
Directory DISK:[SAMPLE]
```

```
QO.LOG:1 77/512
```

```
Total of 1 file, 77/512 blocks.
```

The workaround to this problem is to manually truncate the file using the DCL command SET FILE/TRUNCATE.

This problem has been corrected in Oracle Rdb7 Release 7.0.6. RMU OPTIMIZER_STATISTICS now correctly closes the files created by the /LOG qualifier.

2.3.20 RMU /UNLOAD /AFTER_JOURNAL “/OPTIONS=SHARED_READ” Qualifier

By default, the RMU /UNLOAD /AFTER_JOURNAL command opens input after-image-journal (AIJ) files for exclusive access. This exclusive access is required in order to utilize performance features such as read-ahead (asynchronous prefetch). However, in some situations, this exclusive access is undesirable.

The RMU /UNLOAD /AFTER_JOURNAL command now first attempts to open the AIJ files for exclusive access and, if the file open fails, a second attempt is made to open the file for shared access (allowing other accessors to the AIJ file).

Further, the command line qualifier “/OPTIONS=SHARED_READ” may be specified so that the AIJ files are always opened for shared access. When /OPTIONS=SHARED_READ is specified, no attempt is made to open input AIJ files for exclusive access.

This problem is corrected in Oracle Rdb7 Release 7.0.6.

2.3.21 RMU/COLLECT OPTIMIZER/READ_ONLY Bugcheck

Bug 1368747

The RMU/COLLECT OPTIMIZER_STATISTICS command could produce a bugcheck in PIOUTL\$READ_PAGES when used with the /TRANSACTION=READ_ONLY and /STATISTICS=(CARDINALITY,STORAGE) qualifiers in Oracle Rdb7 Release 7.0.5.

The following example shows a sample RMU COLLECT command line which caused the problem.

```
RMU/COLLECT OPTIMIZER_STATISTICS DATABASE -  
/STATISTICS=(CARDINALITY,STORAGE)/TRANSACTION=READ_ONLY
```

As a workaround to this problem, do not specify /TRANSACTION=READ_ONLY or keep database access to a minimum.

```
RMU/COLLECT OPTIMIZER_STATISTICS DATABASE -  
/STATISTICS=(CARDINALITY,STORAGE)
```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.22 RMU/LOAD Exception in READ_BLOB Routine if Empty Query Header

Bug 1358155

RMU/LOAD would receive an exception in the READ_BLOB routine if the *.UNL unload BRP had begin and end segmented string blob section codes without any blob data. This happened because of an invalid segmented string in the database.

The following example shows the BRP codes from the *.UNL file for a global field query header which caused the problem.

```
NONCORE_BLOB_FLD_DTR_QUERY_HDR - (0)
BEGIN_BLOB_SECTION - (1) : PRESENT
END_BLOB_SECTION - (0)
```

RMU/LOAD now handles this case and does not return an exception.

As a workaround to this problem, redefine the segmented string in the global field definition that has this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.23 RMU/VERIFY/INDEX Always Used Two SORTWORK Files

Bug 1341637

RMU/VERIFY/INDEX always used 2 sort work files for sorting data and index dbkeys to see if they match correctly and did not first check the RDMS\$BIND_SORT_WORKFILES logical before defaulting to two sort work files.

The following example shows the RDMS\$BIND_SORT_WORKFILES logical being defined so that 10 workfiles should be used by VMS SORT. This was ignored by the RMU/VERIFY/INDEX command, which always used only 2 sort work files for the internal sorting it does of index and data DBKEYs to see if they match up.

```
DEFINE RDMS$BIND_SORT_WORKFILES 10
RMU/VERIFY/INDEX=test_index database.rdb
```

Now the RDMS\$BIND_SORT_WORKFILES logical will be checked by RMU/VERIFY/INDEX and only if it is not defined will the default of 2 sort work files be used. Note that since two separate sort streams are used internally by RMU/VERIFY/INDEX, the number of sort work files specified by RDMS\$BIND_SORT_WORKFILES will be used for each stream. Therefore, if 10 sort work files are specified, 20 will actually be created; 10 for the index DBKEY sort stream and 10 for the data DBKEY sort stream.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.24 RMU/ANALYZE/PLACE Gives 0 Path Length for Some Indexes

Bug 1261039

RMU/ANALYZE/PLACE returned a zero path length for hashed indexes which were in different storage areas than their tables, unless the areas where the indexes were located was specified with the /AREA qualifier in addition to the area where the table was located. Only the area where the table is located should be required. The following command returned this zero path length.

```
RMU/ANALYZE/PLACE/OPTION=FULL/AREA=(TABLE_AREA) MF_PERSONNEL_HASH_INDEX
```

This command will now return a correct path length if only the area where the table is located is specified.

To workaroud the problem, specify the area where the hashed index is located in addition to the area where the table is located.

```
RMU/ANALYZE/PLACE/OPTION=FULL/AREA=(TABLE_AREA, INDEX_AREA) -  
MF_PERSONNEL HASH_INDEX
```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.25 RMU/REPAIR/ABM/SPAM Causes Multiple AIP Page Pointers

Bug 1245110

An RMU/REPAIR/ABM/SPAM on a database converted to 7.0 or higher had a problem when reassigning deleted page clusters in uniform storage area SPAM page lists. This problem caused multiple AIP pointers to the same ABM page for different logical areas as well as other inconsistencies.

The following example shows the RMU/CONVERT, RMU/REPAIR and the RMU/VERIFY which detected a number of inconsistencies with the repaired database.

```
RMU/CONVERT database  
RMU/REPAIR/ABM/SPAM database  
RMU/VERIFY/ALL database
```

As a workaroud to the problem, a second RMU/REPAIR/ABM/SPAM command can be executed after the first RMU/REPAIR/ABM/SPAM to fix the inconsistencies.

```
RMU/CONVERT database  
RMU/REPAIR/ABM/SPAM database  
RMU/VERIFY/ALL database  
RMU/REPAIR/ABM/SPAM database  
RMU/VERIFY/ALL database
```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.26 RMU/RESTORE/OPEN_MODE Did Not Set Database Close Mode

Bug 1228099

RMU/RESTORE/OPEN_MODE=AUTOMATIC and RMU/RESTORE/OPEN_MODE=MANUAL did not set the database close mode to the same setting specified for the open mode and was thus inconsistent. Also, RMU/RESTORE did not have an optional /CLOSE_WAIT=n qualifier to specify the number of minutes to wait before automatically closing the database.

The following example shows the RMU/RESTORE command that set the database open mode to automatic, as verified by the RMU/DUMP/HEADER command, but failed to set the database close mode to the same value specified for the open mode.

```
RMU/RESTORE/OPEN_MODE=AUTOMATIC/DIR=DISK:[DIRECTORY] TEST_DB.RBF  
RMU/DUMP/HEADER=PARAMETERS TEST_DB.RDB
```

The database close mode will now be set to the same mode, AUTOMATIC or MANUAL, specified for the database open mode. In addition, if AUTOMATIC is specified for the database open mode, the close mode can optionally be set to TIMED AUTOMATIC by using the new /CLOSE_WAIT qualifier to specify the number of minutes to wait before the database is automatically closed.

```
RMU/RESTORE/OPEN_MODE=AUTOMATIC/CLOSE_WAIT=10/DIR=DISK:[DIRECTORY] TEST_DB.RBF  
RMU/DUMP/HEADER=PARAMETERS TEST_DB.RDB
```

As a workaround to the problem, use SQL ALTER DATABASE to set the database open mode. This will set the database close mode to the same value.

```
SQL> ALTER DATABASE FILENAME TEST_DB OPEN IS  
      AUTOMATIC (WAIT 10 MINUTES FOR CLOSE);
```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.27 RMU/SHOW STATISTICS Bad Results With /RESET/OUTPUT/CLUSTER

Bug 1220433

When RMU/SHOW STATISTICS was used with the /RESET and /OUTPUT and /CLUSTER qualifiers, incorrect statistics were output which were the number of cluster nodes on which the database was opened or a multiple of that number, not the correct statistics for the particular statistics screen.

The following shows that if the database was open on 6 nodes of the cluster the statistics would be the number 6 or a multiple of that number, but only if the /OUTPUT and /RESET qualifiers were used in the same command line.

```
RMU/SHOW STAT TEST_DB/TIME=300/RESET/LOG/OUTPUT=A.DAT/CLUST  
  
                Summary IO Statistics  
name.....    max..... cur..... avg..... count.....  
transactions   600      0      6.0      6
```

To workaround the problem, do not use the /OUTPUT and /RESET commands in the same RMU/SHOW STATISTICS command with the /CLUSTER qualifier.

```
RMU/SHOW STAT TEST_DB/TIME=300/RESET/LOG/CLUST
```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.28 RMU/VERIFY Does Not Report Cause of VMS SORT Error

Bug 1161836

When VMS SORT returned a %SORT-W-SORT_ON error during an RMU/VERIFY of a database because of a disk being full, RMU/VERIFY only returned the %SORT-W-SORT_ON error, not the SYSTEM-W-DEVICEFULL VMS error which was the real cause of the problem.

The following example shows that RMU/VERIFY only returned the %SORT-W-SORT_ON error from VMS SORT if the sort failed because the device the sort work file was on was full, not the -SYSTEM-W-DEVICEFULL and %SORT-E-WRITEERR errors which reveal the actual cause of the problem.

```
$ RMU/VERIFY/ALL/LOG MF_PERSONNEL  
  
%RMU-I-BGNSEGPAG, beginning verification of  
DISK:[DIRECTORY]PERSONNEL.RDB;1 storage area  
  
%SORT-W-SORT_ON, sort or merge routines in incorrect order  
%RMU-F-ABORTVER, fatal error encountered aborting verification  
%SORTS-F-SORT_ON, sort or merge routines in incorrect order  
%RMU-F-FATALOSI, fatal error from the os interface  
%RMU-F-FTL_VER, fatal error for VERIFY operation at 13-JUN-2000 10:43:55.50
```

The following example shows the corrected behavior. Now all errors encountered during the VMS SORT are intercepted and returned by RMU/VERIFY when the sort fails.

```

$ RMU/VERIFY/ALL/LOG MF_PERSONNEL

%RMU-I-BGNSEGPAG, beginning verification of
DISK:[DIRECTORY]PERSONNEL.RDB;1 storage area

%SORT-E-WRITEERR, error writing DISK:[DIRECTORY]SORTWORK1.TMP;1
-SYSTEM-W-DEVICEFULL, device full; allocation failure
%RMU-F-ABORTVER, fatal error encountered; aborting verification
%SORT-F-SORT_ON, sort or merge routines called in incorrect order
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-F-FTL_VER, Fatal error for VERIFY operation at 13-JUN-2000 10:43:55.50

```

To workaroud the problem, make sure the disks used by the VMS SORT WORK files have enough space for any sorting of index and data database keys done by RMU/VERIFY/INDEX.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.29 RMU/VERIFY RMU-E-BADSEGTYT Message Insufficient

Bug 1112631

The RMU/VERIFY RMU-E-BADSEGTYT error message does not give the storage area name, the page, and the line number of the start of the segmented string which has the bad segment type. The following example shows this behavior.

```

RMU/VERIFY/ALL database.rdb
%RMU-E-BADSEGTYT, Storage segment of 8202 found in storage segment header.
Expected segmented string type of -2, found 0.

```

The storage area name, the page, and the line number of the start of the segmented string which has the bad segment type will now be returned by RMU/VERIFY.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.30 RMU/CONVERT Multischema Database Problem

Bug 894941

The conversion of a multischema database from an earlier version of Oracle Rdb to Oracle Rdb V7.0 did not properly set the record length of rows in the RDB\$CATALOG_SCHEMA system table. This caused problems finding database objects during database queries.

With this problem, a SHOW TABLE in SQL will not find the table even though it exists and an SQL CREATE INDEX will fail with the following error:

```
%SQL-F-CATNOTDEF, Catalog {name} is not defined.
```

As a workaround to this problem, do an SQL EXPORT of the database under Oracle Rdb V6.0 and an SQL IMPORT of the database under Oracle Rdb V7.0. If the SQL EXPORT/IMPORT cannot be done, the database should be rebuilt.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.31 RMU/RESTORE From Multiple Tapes Gives False RMU-F-BACFILCOR

Bug 515948

A multi-threaded RMU/RESTORE from multiple tape devices could return a false "%RMU-F-BACFILCOR, Backup file is corrupt error" if a different number of tape drives was used for the restore than was used for the backup. This was a multi-thread problem that depended on both using a different number of master tape devices for the restore than was used for the backup and on how the backed up storage areas were distributed on the tapes. This problem also caused the

continuation tapes to be mounted out of the expected sequence. A single threaded RMU/RESTORE using a single tape device always succeeded and the backed up RBF file was not corrupt.

The following example shows the false RMU-F-BACFILCOR being output in a case where 4 master tape drives were used for the RMU/BACKUP but 2 master tape drives were used for the RMU/RESTORE. It also shows the mounting of the second set of tapes out of order.

```
$ RMU/RESTORE/LOG/NOCDD/NORECOVERY/NOACL/ACTIVE_IO=5/PAGE_BUFFERS=5 -
_.$      /ROOT=TEST_DB -
_.$      /DIRECTORY=DISK:[DIRECTORY] -
_.$      /VOLUMES=4/LOADER_SYNC -
_.$      /LABEL=(TAPE1,TAPE2,TAPE3,TAPE4) -
_.$      $1$MUA101:TEST_DB/MASTER, -
_.$      $1$MUA102:/MASTER
%RMU-I-REXTXT_04, Thread 1 uses devices $1$MUA101:
%RMU-I-REXTXT_04, Thread 2 uses devices $1$MUA102:
%RMU-I-AIJRSTBEG, restoring after-image journal "state" information
%RMU-I-AIJRSTEND, after-image journal "state" restoration complete
%RMU-I-REXTXT_00, Restored root file DISK:[DIRECTORY]TEST_DB.RDB;1
%RMU-I-LOGRESSST, restored storage area DISK:[DIRECTORY]AREA1.RDA;1
.
.
%RMU-I-RESUME, resuming operation on volume 2 using _$1$MUA102
%MOUNT-I-WRITELOCK, volume is write locked
%MOUNT-I-MOUNTED, TAPE2 mounted on _$1$MUA102: (HSJ202)
%RMU-I-RESUME, resuming operation on volume 3 using _$1$MUA101
%RMU-I-RESUME, resuming operation on volume 4 using _$1$MUA102
%RMU-I-READYREAD, mount volume 4 on _$1$MUA102: for reading
Press return when ready:
<- RMU/RESTORE always requests the volume 4 previous to the volume 3.
%MOUNT-I-WRITELOCK, volume is write locked
%MOUNT-I-MOUNTED, TAPE4 mounted on _$1$MUA102: (HSJ202)
%RMU-I-READYREAD, mount volume 3 on _$1$MUA101: for reading
Press return when ready:
%MOUNT-I-WRITELOCK, volume is write locked
%MOUNT-I-MOUNTED, TAPE3 mounted on _$1$MUA101: (HSJ101)
%RMU-F-BACFILCOR, Backup file is corrupt
%RMU-F-FATALERR, fatal error on RESTORE
```

The false RMU-F-BACFILCOR will no longer be returned and the restore will complete correctly.

As a workaround to this problem, use the same configuration of tape drives for the RMU/RESTORE as was used in the RMU/BACKUP or use a single tape drive for the RMU/RESTORE.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.32 RMU/RECLAIM Set SPAM Thresholds Incorrectly on Uniform Areas

Bug 1384559

The RMU/RECLAIM command (see Section 7.2.4) would leave SPAM thresholds set incorrectly when used on a UNIFORM format storage area. This problem did not affect MIXED format storage areas.

When this problem occurs, it is possible to correct the SPAM pages by using the RMU/REPAIR command.

This problem has been corrected in Oracle Rdb7 Release 7.0.6. RMU/RECLAIM will now properly set the SPAM threshold values.

2.3.33 ACCVIOs from RMU/COPY/ONLINE or RMU/BACKUP/AFTER on OpenVMS Alpha V7.1

Bug 1340344

The RMU/COPY/ONLINE and RMU/BACKUP/AFTER commands would sometimes fail with an access violation. For example:

```
$ RMU/COPY/ONLINE/DIR=DEV:[DIR] DB_FILESPEC
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address
=000000007FFA1E58, PC=FFFFFFFF800BD3A8, PS=0000001B

Improperly handled condition, image exit forced.
Signal arguments:  Number = 0000000000000005
                   Name   = 000000000000000C
                   0000000000010000
                   000000007FFA1E58
                   00000000800BD3A8
                   000000000000001B

Register dump:
R0 = 0000000000000009  R1 = 0000000000000000  R2 = 0000000000000003
R3 = 0000000000000090  R4 = 00000000000001FF  R5 = 0000000000000000
R6 = 0000000000D6BB18  R7 = 0000000000000000  R8 = 0000000000000001
R9 = 0000000000E3F000  R10 = 0000000000000200  R11 = 00000000000001FF
R12 = 000000000000FE00  R13 = FFFFFFFF84CCAD70  R14 = FFFFFFFF81025000
R15 = 0000000000000200  R16 = 0000000000000000  R17 = 0000000000000000
R18 = FFFFFFFF8504C078  R19 = 0000000000000008  R20 = 000000000000000E
R21 = 0000000000060878  R22 = 0000000000000040  R23 = 0000000001000000
R24 = 0000000000A0044  R25 = 0000000000000000  R26 = FFFFFFFF800BD3A8
R27 = 0000000000000052  R28 = 0000000000000000  R29 = 000000007FFA1E40
SP = 000000007AF8C0E0  PC = FFFFFFFF800BD3A8  PS = 200000000000001B
```

This problem was caused by an error in the OpenVMS Alpha V7.1 AST code. It is fixed by patch kit ALPSYSA01_071 or later. The OpenVMS release note entry describing the problem is included below:

- An Access Violation may occur at EXE\$AST_RETURN, in an outer mode - most typically user mode - but with the Frame pointer pointing to the kernel stack. The access violation occurs trying to access data on the kernel stack from user mode. This does not crash the system, but causes the user image to exit.

The following URL provides a list of current OpenVMS patch kits.

<http://ftp.support.compaq.com/patches/.new/openvms.html>

2.3.34 RMU/REPAIR/INIT=FREE_PAGES Corrupted Sorted RANKED Indexes

Bug 1491597

RMU/REPAIR/INIT=FREE_PAGES did not work properly with SORTED RANKED indexes. An RMU/VERIFY, done after the repair, showed problems with the SORTED RANKED indexes defined in the repaired database, that were caused by the RMU/REPAIR and the SORTED RANKED indexes had to be rebuilt. RMU/REPAIR now will not cause problems with SORTED RANKED indexes.

The following RMU/REPAIR command caused errors to be returned by RMU/VERIFY for SORTED RANKED indexes defined in the repaired database.

```

RMU/REPAIR/INIT=FREE_PAGES database
RMU/VERIFY/ALL database
%RMU-E-BADDBKFET, Error fetching dbkey 54:8:6
%RMU-W-BTRVFYPRU, B-tree verification pruned at this dbkey
%RMU-I-BTRERPATH, parent B-tree node of 54:8:6 is at 54:3:0
%RMU-I-BTRENTCAR, Inconsistent entry cardinality (C1) of 375 specified
                    for entry 2 at dbkey 54:3:0 using precision of 33.

```

The SORTED RANKED indexes must be rebuilt after the RMU/REPAIR is completed to continue to use the SORTED RANKED indexes with the repaired database.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.3.35 RMU Extract Did Not Process Empty QUERY HEADER Segments

Bug 1368926

In prior releases of Oracle Rdb7, RMU Extract would ignore QUERY HEADER segments that were empty (i.e. zero length). These empty QUERY HEADER segments were used to space the titles on the printed output from tools such as the interactive SQL SELECT statement.

```

SQL> create table QH (
cont> person_name CHAR (20)
cont> query header is 'Name' / '' / 'Person');
SQL> insert into qh value ('Jones');
1 row inserted
SQL> select * from qh;
   Name
-----
 Person
  Jones
1 row selected
SQL> commit;

```

If the query header were defined with all empty segments, then the QUERY HEADER clause was omitted from the output. If only some of the header segments were empty, then RMU Extract could generate an incorrect definition. The following example demonstrates the problem.

```

$ RMU/EXTRACT/ITEM=TABLE sampledb
.
.
.
create table QH (
  PERSON_NAME
  CHAR (20)
  query header is 'Name' / 'Person');
.
.
.

```

As can be seen, the empty segment is omitted from the output. If the empty segment was the first or last segment, then incorrect syntax was generated.

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

2.4 Row Cache Errors Fixed

2.4.1 Row Cache Recovery Began at Too Old Checkpoint With Hot Standby

When the Oracle Rdb7 Row Cache and Hot Standby features were both enabled, it was possible that if the system failed while Hot Standby was in “Catch-Up” mode, a subsequent database re-open would result in the database recovery (DBR) server starting the database recover at a checkpoint location that was excessively old. This, in turn, would cause the DBR to run for a longer time than would have otherwise been required.

This problem has been corrected in Oracle Rdb7 Release 7.0.6. The DBR process now does not consider the checkpoint information for the AIJ Log Catch-Up (LCS) server.

2.4.2 DBR Buffer Count for Node-Failure Recovery With Row Cache

When the Oracle Rdb7 Row Cache feature is enabled, a single database recovery (DBR) server recovers the database after a “node failure” event. Node failure refers to any abnormal database shutdown including system failure (crash or power outage), and DBR, RCS or monitor failure. Because of the amount of recovery work that might be done by this DBR, it is important that enough database page buffers are used to help ensure reasonable performance of the recovery.

The minimum number of database page buffers for the DBR process performing node failure recovery has been increased to 1000 when global buffers are not enabled. When global buffers are enabled, the existing default of the database parameter “NUMBER OF RECOVERY BUFFERS” still applies.

2.5 Hot Standby Errors Fixed

2.5.1 RMU/REPLICATE AFTER_JOURNAL STOP/ABORT Should Close Database

When trying to abort the Hot Standby product using the RMU/REPLICATE AFTER_JOURNAL STOP/ABORT command, there exists a window between the time that Hot Standby shuts down and the database is subsequently closed using the RMU/CLOSE/ABORT command. Within this window, it is possible for users to modify the master database in such a manner as to require re-creating the standby database.

Since a force-exit to close the database is not allowed when Hot Standby is active, it is very desirable to combine the two commands to eliminate the window.

The following example shows the commands required to stop Hot Standby and close the database in a panic situation:

```
RMU/REPLICATE AFTER STOP/ABORT/WAIT/LOG MASTER_DB  
RMU/CLOSE/ABORT=FORCEX/WAIT MASTER_DB
```

There is no work-around to this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.6. The RMU/REPLICATE AFTER_JOURNAL STOP/ABORT command has been enhanced as follows: The /ABORT qualifier now accepts the following optional keyword:

- /ABORT=FORCEX

When you use the FORCEX (forced exit) option, the database is closed and recovery unit journals (RUJ) are recovered and removed.

- /ABORT=DELPRC

When you use the DELPRC (delete process) option, the database is closed, and recovery unit journals (RUJ) are not recovered. The processes and any subprocesses of all database users are deleted.

- /ABORT

When neither DELPRC nor FORCEX is specified, the /ABORT qualifier will maintain its prior behaviour.

Note that the /ABORT=keyword qualifier only operates on the database for which the command was issued. For example, if the /ABORT=DELPRC qualifier is specified on the master database, only the master database processes will be deleted; the standby database will be aborted normally.

Software Errors Fixed in Oracle Rdb7 Release 7.0.5

This chapter describes software errors that are fixed by Oracle Rdb7 Release 7.0.5.

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 Query Using Match Strategy Outline Returns Wrong Results

Bug 974665

The following query returns wrong results when the match strategy outline is used.

```

select  s.proj_code, s.title_code, s.site_code,
        s.scan_ind, s.plan_date, t.bid_date
from    sdp s,
        tcd t
where   s.proj_code      = t.proj_code      and
        s.title_code     = t.title_code     and
        s.site_code      = 'CLEV'          and
        s.scan_ind       = 'Y'             and
        t.bid_date is null;
~S: Outline "QO_ZIGZAG_MATCH" used
Conjunct
Match
Outer loop
  Sort      Conjunct      Index only retrieval of relation SDP
  Index name SDP_IDX [1:1]
Inner loop  (zig-zag)
  Index only retrieval of relation TCD
  Index name TCD_IDX [1:1]
S.PROJ_CODE  S.TITLE_CODE  S.SITE_CODE  S.SCAN_IND
S.PLAN_DATE  T.BID_DATE
          980620259  @@@          CLEV          Y
          9-AUG-1999 00:00:00.00  NULL
1 row selected

```

The NULL predicate column "bid_date" is not part of the index SDP_IDX, which is defined as follows:

```

create index SDP_IDX on SDP
  (SITE_CODE, PLAN_DATE,
   SCAN_IND,  BID_PCT desc,
   PROJ_CODE, TITLE_CODE)
  type is SORTED;
commit work;

```

The outline is defined to use match instead of cross as follows:

```
create outline QO_ZIGZAG_MATCH
id 'CDBFC4B343409886D2FC605C40761388'
mode 0
as (
  query (
    subquery (
      SDP 0 access path index SDP_IDX
      join by match to
      TCD 1 access path index TCD_IDX
    )
  )
)
compliance mandatory
execution options (any);
```

Workarounds for this problem are to use the SQL SET FLAGS 'NOZIGZAG_MATCH' command to turn off zigzag match or to delete the outline.

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

3.1.2 Database Recovery Process Bugchecks at DIOCCHDBR\$UNLATCH_GRCL + 00000398

In very rare cases of process failure when using the row cache feature, it was possible for an Oracle Rdb7 Database Recovery (DBR) to fail with an exception within DIOCCHDBR\$UNLATCH_GRCL (typically at offset 00000398).

This bugcheck was due to an incorrect check while releasing a row cache latch for the failed process.

This problem has been corrected in Oracle Rdb7 Release 7.0.5. The DBR process now correctly validates the hash table slot number.

3.1.3 Dynamic Optimizer Problem with Zigzag Match

In some queries utilizing zigzag match retrieval, a problem with the interaction of the zigzag match and the dynamic optimizer may cause the query to fail to deliver appropriate records.

The queries affected contain a join of two or more tables where the optimizer has chosen to utilize a zigzag match retrieval strategy and dynamic optimization (LEAF) retrieval of data for the inner leg of the match.

The following is an example of the type of strategy associated with the affected queries.

```
Match
Outer loop      (zig-zag)
  Conjunct      Get      Retrieval by index of relation TABLE1
  Index name    TABLE1_INDEX_01 [1:1]
Inner loop      (zig-zag)
  Leaf#02 Sorted TABLE2 Card=128800
  FgrNdx TABLE2_INDEX_01 [0:0] Fan=41
  BgrNdx1 TABLE2_INDEX_02 [0:0] Fan=27
```

A problem in the delivery of data by the inner leg of the match from the dynamic optimizer data buffers prevented the appropriate match records in the outer leg from being found.

A workaround for the problem is to use the RDMS\$DISABLE_ZIGZAG_MATCH logical name or the SQL SET FLAGS statement to disable zigzag match.

```
VMS> define RDMS$DISABLE_ZIGZAG_MATCH 2
or
SQL> set flags 'nozigzag_match';
```

Alternatively, dynamic optimization may be disabled by using the RDMS\$MAX_STABILITY logical name or the SQL SET FLAGS statement:

```
VMS> define RDMS$MAX_STABILITY "TRUE"
or
SQL> set flags 'max_stability';
```

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

3.1.4 DBR Bugcheck in DBR\$RECOVER_RCS Due to AIJ Related Database Shutdown

When the Oracle Rdb7 Row Cache Feature was enabled, the database recovery (DBR) server required that after image journaling was enabled and in a “normal” state. However, in extreme cases, such as after image journaling being shut down due to no more available journals, the DBR process would bugcheck leaving the database unuseable.

This problem has been corrected in Oracle Rdb7 Release 7.0.5. The DBR process is now more tolerant of the AIJ state.

3.1.5 Bugchecks at DIOCCH\$FETCH_SNAP_SEG + 000005C4

In rare cases of process failure when using the row cache feature, it was possible for other processes to later fail with an exception within DIOCCH\$FETCH_SNAP_SEG (typically at offset 000005C4). This problem was discovered during internal testing and was not customer reported.

This bugcheck was due to incorrectly storing a snapshot page pointer in the row cache prior to writing the snapshot page back to the database. If the process failed between these two events, other users of the database could read an invalid snapshot page and this could lead to a bugcheck.

This problem has been corrected in Oracle Rdb7 Release 7.0.5. The snapshot page pointer in the row cache is not updated until the snapshot page has been written back to disk.

3.1.6 Random Corrupt Pages on Fast Processors

Bug 1142549

In rare circumstances, spurious checksum errors were being reported by Oracle Rdb. An obscure timing issue was found involving the Rdb Global Buffer feature when multiple processes attempted to access the same database pages at the same time. This error was more prevalent with the faster processors. In Oracle Rdb7 Release 7.0.3.1 and above, when this error occurs, an automatic re-read of the page obtains the correct data, and processing continues.

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

3.1.7 Wrong Results from 3-Way Join Using Cross/Zigzag_Match

Bug 1041071

Wrong results were being returned from a 3-way join using the cross/zigzag match strategy.

```
select  t1.join_id,
        t2.ctrct_rvs_seq_no,
        t3.ctrct_expr_dte
  from t1, t2, t3
 where t1.id = 'V380025A' and
        t1.col2 = '01' and
        t1.col3 = 'Y' and
        t1.join_id = t2.join_id and
        t1.id = t2.id and
        t1.join_id = t3.join_id;
```

where the following indexes are defined :

```
create index t1_ndx on t1 (j_id, t1_col2, t2_col3, join_id)
create index t2_ndx on t2 (j_id, join_id, t2_col3)
create index t3_ndx on t3 (join_id)
```

The key parts of this query which contributed to the situation leading to the error are these:

1. t1, t2, and t3 are joined by one common segment, join_id
2. join_id is the leading segment in t3_ndx, 2nd segment in t2_ndx, and the 4th segment in t1_ndx
3. t1 and t2 are joined by j_id and join_id columns, which are leading contiguous segments in t2_ndx, but separated by 2 segments in t1_ndx
4. 2nd and 3rd segment of t1_ndx are used as equality predicates with constant values

The workaround is to define the logical RDMS\$SET_FLAGS, or use the SQL SET FLAGS statement with the value NOZIGZAG_MATCH or define the logical name RDMS\$DISABLE_ZIGZAG_MATCH as "2".

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

3.1.8 Query Slowdown Caused by Subquery With MIN/MAX Functions

Bug 907429

A query that used to take 40 seconds to run under Rdb 6.1 and Rdb 7.0.1.1 now takes 4 hours to run under 7.0.1.2 through 7.0.2.1.

The following query is the example of the problem query, where the subqueries have aggregate functions MIN and MAX.

```
select distinct(f1.number), f1.name, f1.address, f1.dbkey from foo f1
  where (select min(f2.name) from foo f2
         where f1.number = f2.number) <>
        (select max(f3.name) from foo f3
         where f1.number = f3.number)
 order by f1.number;
```

There is no good workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

3.1.9 GROUP BY/HAVING Query From a View With LIMIT TO Clause Returns Wrong Results

Bug 1204964

The following GROUP BY/HAVING query from a view with LIMIT TO clause returns wrong results.

```
SELECT seanic_month, COUNT(*) FROM A_2_view
  GROUP BY seanic_month HAVING seanic_month = '1999 03';
Aggregate      Conjunct      Firstn Conjunct      Get
Retrieval sequentially of relation TABLE_2
SEANIC_MONTH
1999 03                12
1 row selected
```

where A_2_view is defined as:

```
create view A_2_VIEW
  (SEANIC_MONTH) as
  select
    C1.SEANIC_MONTH
  from table_2 C1
  where ((C1.SOURCE_TABLE_TYPE = 'MY')
    and (C1.MONTH_END_FLAG = 'Y'))
  order by C1.SEANIC_MONTH desc
  limit to 36 rows;
```

The key parts of this query which contributed to the situation leading to the error are these:

1. The main select query has GROUP BY/HAVING clause
2. The view is defined as a select query with LIMIT TO clause

The workaround is to remove the LIMIT TO clause from the view query.

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

3.1.10 Query Returns Wrong Results When the Sequence of Same Context Predicates is Broken Up

Bug 1222168

The following query returns wrong results (should be 0 rows):

```

select h.blue
  from honda h where
    h.yellow = (select t.yellow from toyota t limit to 1 row) and
    h.purple = 'some-color' and
    not exists (select * from animal a inner join mammal m
               using (donkey, horse, pony)
               where
                 m.color = h.color) and
    h.blue <> 0;
Cross block of 3 entries
Cross block entry 1
  Aggregate      Firstn  Index only retrieval of relation toyota
  Index name     toyota_IDX1 [0:0]
Cross block entry 2
  Leaf#01 FFirst honda Card=439
  BgrNdx1 honda_IDX1 [0:0] Bool Fan=13
Cross block entry 3
  Conjunct      Aggregate-F1  Conjunct
  Match
  Outer loop
  Sort  Conjunct      Conjunct
  Index only retrieval of relation animal
  Index name  animal_IDX5 [0:0]
  Inner loop  (zig-zag)
  Conjunct   Index only retrieval of relation mammal
  Index name  mammal_IDX1 [0:0]

```

The key parts of this query which contributed to the situation leading to the error are these:

1. Two or more predicates on the same table, for example, T1
2. Followed by NOT EXISTS predicate on different tables, for example, T2 and T3
3. Followed by another predicate on T1

The workaround is to define the logical RDMS\$SET_FLAGS, or use the SQL SET FLAGS statement with the value MAX_STABILITY or define the logical name RDMS\$MAX_STABILITY, or group all the predicates on T1 together instead of having them separated by another predicate of a different context.

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

3.1.11 Wrong Results When GROUP BY Columns are NOT Leading Subset of UNION Columns

Bug 1177495

The following GROUP BY query returns wrong results when GROUP BY columns are NOT leading subset of UNION columns:

```

SELECT  COURSE_NO, CLASS_NO, COUNT(*)
FROM
  (SELECT  Q.STDT_ID, Q.COURSE_NO, Q.CLASS_NO
   FROM
     (SELECT  STDT_ID,
              REQ_COURSE_NO AS COURSE_NO,
              REQ_CLASS_NO_1 AS CLASS_NO
        FROM    NH_TEMP A
      UNION
      SELECT  STDT_ID,
              REQ_COURSE_NO AS COURSE_NO,
              REQ_CLASS_NO_2 AS CLASS_NO
        FROM    NH_TEMP A
     ) AS Q, IN_TEMP B, PROG_TAB C
   WHERE
     B.STDT_ID = Q.STDT_ID AND
     C.PROG_ID = B.PROG_ID AND
     C.OPT_ID = B.OPT_ID) AS X
GROUP BY COURSE_NO, CLASS_NO;

```

The key parts of this query which contributed to the situation leading to the error are these:

1. Select query with GROUP BY clause from a subquery with UNION legs
2. GROUP BY column order starts from the 2nd column of UNION column order

The workaround is to use UNION ALL instead of UNION.

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

3.1.12 New Index Scan Algorithm Not Effective With Some Sorted Indices

Bug 1164982

In prior versions of Oracle Rdb7 (since release V7.0.1.3), the logical name RDMSSINDEX_PART_CHECK could be defined to "1" to enable a new end-of-partition checking algorithm. This new algorithm can greatly improve concurrency and performance when scanning partitioned sorted indices. The new algorithm avoids reading neighbor partitions to determine the end condition of the scan and therefore allows concurrent table processing by partitioned applications (see the PARTITION clause of the SET TRANSACTION ... RESERVING statement in the SQL Reference Manual).

However, the new algorithm was not being used when the partition USING clause listed a subset of the columns of the index. An example of such an index is shown here:

```

create unique index PERSONS_IDX
  on PERSONS (
    LAST_NAME,
    FIRST_NAME,
    MIDDLE_INITIAL
  )
  type is SORTED
  store
    using (LAST_NAME)
      in EMPIDS_LOW
        with limit of ('Dement')
      in EMPIDS_MID
        with limit of ('Myotte')
      otherwise in EMPIDS_OVER;

```

In this example, the USING clause uses only one of the columns of the index.

This problem has been corrected in Oracle Rdb7 Release 7.0.5. This release of Oracle Rdb7 now adapts correctly to this type of index definition.

Note

This new algorithm is the default behaviour in the next major release of Oracle Rdb. In that release the RDMSSINDEX_PART_CHECK algorithm is only used to disable the algorithm.

3.1.13 Wrong Results From a View Query With Left Outer Join and SUBSTRING Function

Bug 1247379

The following select query from a view with left outer join and the SUBSTRING function should return 1 row:

```
select * from V1 where VCOL2 = 'abc';

where V1 is defined as :
create view V1 (VCOL1, VCOL2, VCOL3) as
select  A1.COL1,
        substring(A1.COL2 from 1 for 3),
        A2.COL1
from
    (select COL1,COL2 from TAB1) as A1
left outer join
    TAB2 as A2
    on A2.COL1 = A1.COL1 ;
0 rows selected
```

The key parts of this query which contributed to the situation leading to the error are these:

1. A view query joining 2 tables with left outer join
2. One of the view columns is a SUBSTRING function
3. The outer join table is empty
4. The selection predicate uses the column with the SUBSTRING function

The workaround is to redefine the view as a derived table, as in the following example.

```
select * from
    (select  A1.COL1,
            substring(A1.COL2 from 1 for 3),
            A2.COL1
    from
        (select COL1,COL2 from TAB1) as A1
    left outer join
        TAB2 as A2
        on A2.COL1 = A1.COL1)
    as V1 (VCOL1, VCOL2, VCOL3)
where VCOL2 = 'abc';
```

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

3.1.14 Query With EXISTS and SUBSTRING Bugchecks

Bug 1162094

The following query with EXISTS and SUBSTRING bugchecks:

```
attach 'file personnel';

create index EMP_STATUS_CODE
on EMPLOYEES(STATUS_CODE);

sel * from employees e
  where exists
    (select * from candidates c
     where e.STATUS_CODE = substring (c.CANDIDATE_STATUS from 1 for 1)
    );
```

The key parts of this query which contributed to the situation leading to the error are these:

1. Select query from a table with "exists" clause where the tables are joined using the equality predicate
2. The 1st table column has an index defined and the 2nd table column has no index
3. The 2nd table column has a substring function
4. The optimizer uses MATCH strategy instead of CROSS

The workaround is to define the logical RDMS\$SET_FLAGS, or use the SQL SET FLAGS statement with the value NOZIGZAG_OUTER (or NOZIGZAG_MATCH).

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

3.1.15 Memory Leak for Trigger Actions

Bug 1229610

In prior releases of Oracle Rdb, a small memory leak (that is allocated memory that is not released until image rundown) occurred when using domain CHECK constraints and INSERT/UPDATE statements in trigger actions.

This problem would only be noticed if the application attached to and disconnected from Rdb databases frequently or the process ran a server process that attached and disconnected often.

For the problem to occur the following must be true:

- An INSERT, DELETE, or UPDATE statement must cause a trigger to be activated
- The trigger must include an INSERT or an UPDATE on a table
- The table columns affected by these statements must be based on a domain with either an explicit domain CHECK constraint defined (possibly inherited from an Oracle CDD/Repository VALID IF definition), or an implicit precision check created for DECIMAL and NUMERIC data types when the dialect is SQL92 or ORACLE LEVEL1.

Note

This problem does not occur for column and table CHECK constraints. That is, CHECK constraints defined using the CREATE/ALTER TABLE statement.

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

3.2 SQL Errors Fixed

3.2.1 Incorrect Output in SHOW STORAGE AREA (USAGE) Display

In prior versions of Oracle Rdb, the SHOW STORAGE AREA (USAGE) output did not include tables which had storage maps without STORE clauses. This meant that tables which were implicitly mapped to the default storage area were not displayed.

Additionally, global and local temporary tables, which are maintained in virtual memory and therefore not mapped to any storage area, were incorrectly listed as being mapped to the default storage area.

The following example shows these problems. In this example, the temporary table GLOBAL_TEMP_0 is listed incorrectly, and the base table BASE_TABLE_1, which should be displayed, is not.

```
SQL> create global temporary table GLOBAL_TEMP_0 (a integer)
cont>     on commit preserve rows;
SQL>
SQL> create table BASE_TABLE_1 (a integer);
SQL> create storage map BASE_TABLE_1_MAP for BASE_TABLE_1
cont>     disable compression;
SQL>
SQL> show storage area (usage) *
Storage Areas in database with filename USAGE

Database objects using Storage Area RDB$SYSTEM:
Usage          Object Name                               Map / Partition
-----
Default List Area
Default Area
Table          GLOBAL_TEMP_0                             (no map)
SQL>
```

These problems have been corrected in Oracle Rdb7 Release 7.0.5. The SHOW STORAGE AREA (USAGE) display no longer includes temporary tables and now includes all implicitly mapped tables. The notation "(no store)" in the output indicates that the storage map implicitly maps the table to the default storage area.

The following examples show the revised output for the same database:

```
SQL> show storage area (usage) *
Storage Areas in database with filename USAGE

Database objects using Storage Area RDB$SYSTEM:
Usage          Object Name                               Map / Partition
-----
Default List Area
Default Area
Storage Map    BASE_TABLE_1                             BASE_TABLE_1_MAP (no store)
SQL>
```

3.3 Oracle RMU Errors Fixed

3.3.1 RMU/BACKUP/AFTER/EDIT_FILE Keyword "YEAR" is Producing a Value of 1999

Bug 1135825

When using edit strings for the AIJ backup filename, and trying to get the year in the filename, the year is producing a value of "1999", but the year is "2000".

This problem *only* occurs on January 1, 2000. The year is correctly produced for December 31, 1999 as well as January 2, 2000 and all future dates.

The following example shows how to reproduce this problem, when the system date is set to January 1, 2000:

1. Create an MF_PERSONNEL database
2. Issue the following commands:

```
$ rmu/set after_journal/enable/reserve=5 -  
  /backup=automatic -  
  /add=(name=aij1, file=task$dka0:[sqluser70.dg.aij.aij_1]aij1, -  
    backup_file=task$dka0:[sqluser70.dg.aij.aij_1]aij1_bck, -  
    edit_filename=(YEAR,MONTH,DAY_OF_MONTH,"-",SEQUENCE)) -  
  /add=(name=aij2,file=disk$user:[dir]aij2, -  
    backup_file=disk$user:[dir]aij2_bck, -  
    edit_filename=(YEAR,MONTH,DAY_OF_MONTH,"-",SEQUENCE)) -  
  /add=(name=aij3,file=disk$user:[dir]aij3, -  
    backup_file=disk$user:[dir]aij3_bck, -  
    edit_filename=(YEAR,MONTH,DAY_OF_MONTH,"-",SEQUENCE)) -  
  /add=(name=aij4,file=disk$user:[dir]aij4, -  
    backup_file=disk$user:[dir]aij4_bck, -  
    edit_filename=(YEAR,MONTH,DAY_OF_MONTH,"-",SEQUENCE)) -  
  mf_personnel
```

3. Create a table called A:

```
SQL> CREATE TABLE A (COLA char(3), COLB char(8));
```

4. Force an AIJ journal switch-over

```
$ RMU/SET AFTER/SWITCH MF_PERSONNEL
```

5. Then check the AIJ backup filename for AIJ_1 once it has filled up:

```
TASK> dir disk$user:[dir]*.aij  
  
Directory disk$user:[dir]  
AIJ1_BCK19990101-0.AIJ;1  
  
Total of 1 file.
```

There is no workaround to this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.5.

3.3.2 RMU/SHOW STATS "Average Per Transaction" is Relative to Epoch

The RMU Show Statistic utility displays the "Average Per Transaction" information relative to the statistics epoch, which is when the database was originally opened for statistics collection. However, it is often desirable to display information based on recent statistics collection. This is not currently possible.

The only workaround is to close and re-open the database, which is not recommended.

This problem has been corrected in Oracle Rdb7 Release 7.0.5. The RMU Show Statistic utility has been enhanced to allow the “Average Per Transaction” information to be displayed based on the last 30 statistics collections. This is known as a running average. It should be noted that the running average is computed using non-zero values (just as the normal average is computed). This means that the running average reflects the average of the most recent 30 periods of activity.

The running average display can be selected using the Tools menu, obtained using the “!” keystroke. Selecting the “Display running rate-per-sec avg” option will display the running average information. Selecting the “Display overall rate-per-sec avg” will return the display to normal. The running average display is also available during the replay of a binary input file. The screen headings will display the selected option appropriately.

3.4 Hot Standby Errors Fixed

3.4.1 Hot Standby Performance Impact on Master Database is Substantial

The Hot Standby product performance impact on the master database, and the application processing on the master database, is significantly noticeable. When measuring AIJ throughput (“AIJ blocks written”), the impact of using Oracle Rdb7 Release 7.0.3.1 Hot Standby with the “Cold” synchronization mode is approximately 35% of that when Hot Standby is not active, and the “Commit” synchronization mode is approximately 65% degradation.

This problem has been corrected in Oracle Rdb7 Release 7.0.5. Substantial performance analysis has been performed, both in controlled laboratory environments and real-world customer environments. The goal of this analysis was to identify bottlenecks and configurations that impact application processing when Hot Standby is active. As a result of this analysis, several Hot Standby algorithms, as well as general database algorithms, have been enhanced.

These enhancements result in substantial Hot Standby performance improvements. When measuring AIJ throughput (“AIJ blocks written”), the impact of using Oracle Rdb7 Release 7.0.5 Hot Standby with the “Cold” synchronization mode is approximately 5% of that when Hot Standby is not active, and the “Commit” synchronization mode is approximately 30% degradation.

Note that, with Oracle Rdb7 Release 7.0.5, the Hot Standby “Commit” synchronization mode performs *better* than the Oracle Rdb7 Release 7.0.3.1 “Cold” synchronization mode (30% versus 35%).

Software Errors Fixed in Oracle Rdb7 Release 7.0.4

This chapter describes software errors that are fixed by Oracle Rdb7 Release 7.0.4.

4.1 Software Errors Fixed That Apply to All Interfaces

4.1.1 RMU/LOAD into Temporary Table

Previously, RMU would not allow loading into a temporary table. While in many cases loading into a temporary table would have little value, using triggers on a temporary table may make this an attractive capability.

This restriction has been lifted in Oracle Rdb Release 7.0.4. Oracle Rdb RMU/LOAD will now load data into temporary tables. Note that the contents of the temporary table are available only to the RMU/LOAD process and will disappear when the RMU/LOAD operation completes.

4.1.2 Divide by Zero Error in Query on Large Table

Bug 800006

A simple query on a large table resulted in the following error.

```
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-SYSTEM-F-FLTDIV_F, arithmetic fault, floating divide by zero at
PC=00375C39, PSL=03C00000
```

The following is an example of the conditions needed to cause the error and the simple query used to evoke the error.

```
create table MIS_RTLSAL (RETL_CODE char(4), RETL_CREDITS integer);
create unique index MIS_RTLSAL_00 on MIS_RTLSAL (RETL_CODE);
commit;
select * from mis_rtlsal limit to 1 row;
```

For the case in which this problem was reported, the cardinality of the table was 161733114 rows and the row cluster factor for the table was 0.2081383. The large table cardinality was one of the key contributing factors. The error occurred in the Rdb Optimizer logic as it was trying to compute the cost of retrieving rows from the database.

As a workaround, this problem can be avoided by using the old cost model for the Rdb Optimizer. You can enable use of the old cost model, for example in interactive SQL, by entering the statement `SET FLAGS 'OLD_COST_MODEL';`

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.1.3 Wrong Results With COUNT DISTINCT CASE

Bug 763963

The following query has two grouped aggregate value columns (indicated by the COUNT operation and the GROUP BY clause), a project operation (DISTINCT), and a CASE clause. These are the key factors contributing to the problem.

```
select ndate, node,
       count(distinct(case device when 'NETWORK' then process_id else null end))
       as NET_DEV_COUNT,
       count(distinct(case device when '' then process_id else null end))
       as NUL_DEV_COUNT
from rdb_usage group by ndate, node, product
order by ndate desc;
```

With two or more such grouped aggregate columns, the query would return wrong results for all but one of the aggregate columns. With only one grouped aggregate column in the query, the results returned were correct.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.1.4 Bugcheck at RDMS\$\$RDMSHEMA_UNLOAD_META+40 on Drop Area With Cascade

Bug 1022562

A problem with the way memory was allocated during the removal of a storage area caused local memory to be incorrectly overwritten which resulted in an access violation at RDMS\$\$RDMSHEMA_UNLOAD_META+40.

This problem was mainly seen when at least one table spanned two or more storage areas including the storage area being dropped.

The following is an example of the storage map and alter database statement which may show this problem. In the example, a storage map is created for a table for storage across two storage areas.

```
SQL> create storage map tabl_map for tabl
cont> store using ( coll )
cont> in data_1 with limit of ( 1000 )
cont> in data_2 with limit of ( 2000 )
cont> ;
```

If, sometime later, the database is altered to drop one of these storage areas, an access violation may occur.

```
SQL> alter database file testdb
cont> drop storage area data_1
cont> cascade;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP
%SQL-I-BUGCHKDMP, generating bugcheck dump file SQLBUGCHK.DMP
%SYSTEM-F-ACCVIO, access violation, reason mask=01, virtual
address=EF9A4AC2 ...
```

A possible workaround for this problem is to alter the appropriate storage maps to exclude the storage area you wish to drop prior to altering the database. See the example below.

```

SQL> alter storage map tabl_map
cont> store using ( coll )
cont> in data_2 with limit of ( 2000 ) reorganize;
SQL> alter database file testdb
cont> drop storage area data_1
cont> cascade;

```

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.1.5 Unexpected I/O During DROP and TRUNCATE TABLE

Bug 989292

If a table contained one or more columns of LIST OF BYTE VARYING type, then the DROP TABLE and TRUNCATE TABLE statements would execute the equivalent of the DELETE FROM table DML statement to erase the list data from the database.

Unfortunately this meant that these statements updated the indices of the table and therefore performed unnecessary I/O to the database and journal files. In addition to this problem, TRUNCATE TABLE erroneously executed BEFORE and AFTER TRIGGER actions and some integrity constraints.

This problem has been corrected in Oracle Rdb7 Release 7.0.4. Oracle Rdb now uses a different mechanism to erase the list data which no longer causes updates to the indices, or constraint and trigger execution. The result is a significant reduction in I/O for tables containing list data. There is no change in behavior for tables that do not contain LIST OF BYTE VARYING columns.

Oracle recommends that SET TRANSACTION ... RESERVING be used to lock the table for EXCLUSIVE WRITE mode to reduce I/O, CPU and virtual memory usage during these operations. If possible, attaching to the database using the RESTRICTED ACCESS clause will further reduce I/O to the snapshot file (SNP) for the LIST STORAGE AREA. Testing of the revised algorithm for DROP TABLE showed a reduction of 10% in asynchronous reads, 82% in synchronous reads, 47% in asynchronous writes and 90% in synchronous writes when comparing the operations.

The first script uses the default reserving mode of SHARED WRITE. This will force all changes to the table to be logged to the snapshot file, and require the Rdb Server to perform row locking (or at least maintain data structures to support row locking).

```

SQL> attach 'file TEST';
SQL> set transaction read write;
SQL> drop table EMPLOYEES cascade;
SQL> commit;

```

The second script uses EXCLUSIVE WRITE to avoid the snapshot I/O for the EMPLOYEES row changes, and RESTRICTED ACCESS to eliminate snapshot I/O for the LIST storage area.

```

SQL> attach 'file TEST restricted access';
SQL> set transaction read write reserving EMPLOYEES for exclusive write
SQL> drop table EMPLOYEES cascade;
SQL> commit;

```

The reduced I/O, CPU usage and virtual memory requirements contributed to a significant reduction in elapsed time for both DROP TABLE and TRUNCATE TABLE when the table contained LIST OF BYTE VARYING columns. Improvements on specific databases will depend on database design, quantity

of data, and over all system resources and therefore may vary from those reported from the Oracle Rdb test environment.

4.1.6 Incorrect Rounding of Negative Numbers in the Round Function

The Round function in SQL\$FUNCTIONS.EXE or SQL\$FUNCTIONSnn.EXE incorrectly rounds negative numbers. This problem has been fixed.

For example, round of (-1.56, 0) would round to -1.0 Not -2.0.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.1.7 Ignored Join Order Led to Poor Query Performance

The following query executed in 1 second with Oracle Rdb7 Release 7.0.1.4. In some later version (tested using Oracle Rdb7 Release 7.0.3.1), the query completed after 9 minutes. Here is the query.

```
select ZM.TEISEI_KGO, PM.PM_ST, PM.OK_ST
  from
    PM, PM_ZUMEN PZ, ZUMEN ZM
  where
    PM.HINBAN      = '009627401' and
    PZ.HINBAN      = PM.HINBAN and
    ZM.ZUMEN_NO    = PZ.ZUMEN_NO and
    ZM.VER         = PZ.VER and
    ZM.TEISEI_KGO = (select max(TEISEI_KGO) from ZUMEN ZM2
                     where ZM2.ZUMEN_NO = ZM.ZUMEN_NO and
                           ZM2.VER      = ZM.VER);
```

Using interactive SQL, for example, one could compare the Optimizer query strategy and cost estimates by entering the SET FLAGS 'STRATEGY,ESTIMATE' statement before executing the query. The cost estimate for the V7.0.1.4 query strategy was less than that of the V7.0.3.1 chosen strategy. The good strategy joined the tables in the following order.

PZ.PM_ZUMEN - ZM.ZUMEN - ZM2.ZUMEN - PM.PM

The poor strategy joined the tables in the following order.

ZM.ZUMEN - ZM2.ZUMEN - PZ.PM_ZUMEN - PM.PM

The Optimizer under Oracle Rdb7 Release 7.0.3.1 was ignoring the good join order. That is, the optimizer did not consider the good join order as providing a possible solution for the query strategy.

As a workaround, you can force Rdb to use the good query solution by creating a query outline under Oracle Rdb7 Release 7.0.1.4 or earlier and then applying that outline to Oracle Rdb7 Release 7.0.3.1.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.1.8 GROUP BY Query on a Distinct Subquery Returns Wrong Results

Bug 1089991

The following query, using match strategy, returns the wrong results.

```

select n1, n5, count(*)
  from
    (select distinct C1.N1, C1.N2, C1.N3, C2.N5
     from T1 C1, T2 C2
     where (C1.N3 = C2.N4))
    as v1 (n1, n2, n3, n5)
  group by n1,n5;
Aggregate
Merge of 1 entries
Merge block entry 1
Reduce Sort   Conjunct
Match
Outer loop
Sort   Get      Retrieval sequentially of relation T2
Inner loop
Temporary relation   Sort   Get
Retrieval sequentially of relation T1
N1           N5
val2         val5           1
val3         val5           1
vall1        val5           1
val4         val5           1
vall1        val5           1
val2         val5           1
6 rows selected

```

Where t1 and t2 are defined as follows:

```

create table T1 (
  N1 CHAR (12),
  N2 INTEGER,
  N3 CHAR (4));

create table T2 (
  N4 CHAR (4),
  N5 CHAR (12));

commit work;

insert into T1 value ('val2', 1001, '5124');
insert into T1 value ('val3', 1002, '5124');
insert into T1 value ('vall1', 1003, '5159');
insert into T1 value ('val2', 1004, '5159');
insert into T1 value ('vall1', 1005, '5163');
insert into T1 value ('val2', 1006, '5163');
insert into T1 value ('vall1', 1007, '5152');
insert into T1 value ('val2', 1008, '5152');
insert into T1 value ('vall1', 1009, '5144');
insert into T1 value ('val4', 1009, '5144');

insert into T2 value ('5124', 'val5');
insert into T2 value ('5163', 'val5');
insert into T2 value ('5144', 'val5');

```

This problem is introduced by the redundant sort elimination enhancement made in an earlier release of Oracle Rdb7. The Optimizer eliminates the GROUP BY sort as redundant as follows.

```

By combining the GROUP BY sort (C1.N1, C2.N5) and
DISTINCT sort (C1.N1, C1.N2, C1.N3, C2.N5) into
(C1.N1, C2.N5, C1.N2, C1.N3)

```

Later the match strategy, using the join column (C1.N3 = C2.N4), changes into (C1.N3, C2.N5, C1.N2, C1.N1) and thus produces the wrong order for the GROUP BY operation.

The fix restores the GROUP BY sort to produce the correct result.

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.1.9 After Image Journal File Format Change

With the new support for the LogMiner(tm) for Oracle Rdb feature, the After Image Journal (AIJ) internal file format minor version number has been updated for Release 7.0.4. If you enable the LogMiner for Oracle Rdb, After Image Journal files created by this version of Oracle Rdb7 may not be accepted by prior versions of Oracle Rdb7.

For this reason, you should make certain to verify and then backup your database(s) and AIJ file(s) before upgrading to Oracle Rdb7 Release 7.0.4.

4.1.10 ORDER BY Ignored in Query With a Sub-select Statement

Bug 1073357

The following query, having an explicit ORDER BY clause, would return rows in the wrong order.

```
select u.user_id, u.user_full_name,
       (select group_id from user_group_usgr gr
        where gr.user_id = u.user_id and gr.user_id = 'LEAM')
  from user_user u
 order by u.user_id;
```

The key parts of this query which contributed to the situation leading to the error are these:

1. an ORDER BY clause for the outer select statement
2. a sub-select statement with its own WHERE clause
3. a portion of the WHERE clause in the form: column = 'literal-value' (in this example: gr.user_id = 'LEAM')
4. the referenced column (gr.user_id) is the same, though possibly from a different table, as the one named in the ORDER BY clause (u.user_id).

Given these conditions, past versions of Oracle Rdb would ignore the ORDER BY clause. Oracle Rdb assumed that the ordering was being done on a single value (in this case the value 'LEAM'). There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.1.11 Query With Sort/Forward Scan Instead of Reverse Scan Slows Down

Bug 901904

The following query slows down drastically in Oracle Rdb7 due to the Sort /Forward strategy as compared to Oracle Rdb V6.1 where the reverse scan is applied.

```
select * from t1 where
       c4 >= 500000 and
       c1 = '10' and
       c2 = '460' and
       c3 = '01'
 order by c4 desc;
Firstn Sort
Leaf#01 BgrOnly T1 Card=9022
BgrNdx1 T1_NDX [1:0] Bool Fan=12
```

The following is the strategy output in Oracle Rdb V6.1.

```
Firstn Conjunct      Get      Retrieval by index of relation T1
Index name T1_NDX [1:0] Bool      Reverse Scan
```

The table and index are defined as follows:

```
create table T1 (
  tsn integer,
  c1 char (2),
  c2 char (3),
  c3 char (2),
  c4 integer);

create index T1_NDX on T1 (c4, c1, c2, c3);

commit work;
```

Oracle Rdb7 uses the new cost model where the index scan cost increases significantly (in the range of 10 to 20 times compared to old cost model) in order to reflect the more accurate rate of I/O retrievals.

Reverse scan overhead cost depends on the forward scan index cost since it is estimated as 10% of that cost, but sort cost is estimated based on the cardinality of the tables and some startup fixed cost.

Consequently, the query will select sort and forward scan strategy over reverse scan since sort cost becomes less expensive than reverse scan and thus the sort suffers performance degradation at run time.

This fix may have a wide impact on other queries where the match with a combination of sort is chosen over the cross. This fix may now revert the query back to cross strategy due to the more expensive costing of sort.

The workaround would be to use the old cost model.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.1.12 Query With Selection Predicates Over UNION Legs Returns Wrong Results

Bug 1030588

The following view query with selection predicates should return 0 rows.

```
select count(*) from v1
  where
    trade_date = 36527 and
    currency    = 'EUR' ;
Aggregate      Reduce Sort
Merge of 2 entries
Merge block entry 1
Conjunct
Match
Outer loop
Sort Get Retrieval sequentially of relation T1 <=== missing conjunct
Inner loop
Get      Retrieval by index of relation T2 <=== missing conjunct
Index name T2_IDX [0:0]
Merge block entry 2
Conjunct      Get      Retrieval sequentially of relation T1
2
1 row selected
```

Where the view v1 is defined as follows:

```
create view v1 (trade_date, currency) as
  select      p.settle_date, c.currency
    from      T1 c, T2 p
   where      (c.tradenum = p.tradenum)
 union
  select      trade_date, currency
    from      T1
  ;
```

A fix for bug 548011 was made in Oracle Rdb7 Release 7.0.1.6 to push down the selection predicates into the union legs but the fix introduced this problem.

The above query should push the predicates "trade_date = 36527" and "currency = 'EUR'" into the Merge blocks (union legs).

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.1.13 Left Outer Join View Query With CASE Statement Returns Wrong Results

Bug 1033975

The following left outer join view query with CASE statement returns the wrong result (0 rows).

```
select buys, trade_date, update_type, portfolio, region
  from view1 where
    region = 'E' and
    portfolio = 'JOHNE'
    and buys > 0;
Reduce Sort      Conjunct
Cross block of 2 entries      (Left Outer Join)
  Cross block entry 1
    Conjunct
  Cross block of 2 entries      (Left Outer Join)
    Cross block entry 1
      Conjunct      Get
      Retrieval by index of relation T1
      Index name  T1_NDX [1:1] Bool
    Cross block entry 2
      Conjunct      <=== this extra conjunct is causing the problem
    Merge of 1 entries
      Merge block entry 1
      Conjunct      Index only retrieval of relation T2
      Index name  T2_NDX [1:1]
    Cross block entry 2
      Conjunct      Conjunct
      Index only retrieval of relation T3
      Index name  T3_NDX [1:1]
0 row selected
```

Where view1 is defined as follows:


```

create view view1 (
    region,
    trade_date,
    update_type,
    portfolio,
    buys ) AS
select
    c2.region,
    c2.trade_date,
    c2.update_type,
    c2.portfolio,
    c2.buys
    from view2 as c2
    left outer join
    t3 as c3 on (c2.region = c3.region)
group by
    c2.region,
    c2.trade_date,
    c2.update_type,
    c2.portfolio,
    c2.buys ;

```

and view2 is defined as follows:

```

create view view2 (
    region,
    trade_date,
    update_type,
    portfolio,
    buys ) as
select
    c1.region,
    c1.trade_date,
    c1.update_type,
    c1.portfolio,
    case
        when (c1.recommendation > c1.held_when_recommended)
        then c1.recommendation
        else 0
    end
    from t1 as c1
    left outer join
    (select c5.region, c5.trade_date
     from t2 c5) as c4 ( f1, f2)
    on (c1.region = c4.f1) ;

```

In a previous release of Oracle Rdb7, a fix for bug 767931 was included where the extra conjunct was generated for a left outer join query. This is usually not a problem except when a CASE statement is used in the 2nd view to further qualify the column of the selection predicates as shown above.

In the example, the conjunct of "buys" selection predicate requires the table T1 and correctly generates it in the 1st leg of the left outer join query but then it incorrectly generates it again in the 2nd cross leg where only the T2 table is available.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.1.14 Query Slower Using Cross Strategy and Outline Fails to Restore to Match

Bugs 1066620 and 1066599

The following query once used a match strategy and performed well. After Oracle Rdb7 Release 7.0.1.4, the strategy changed to a cross and the query ran much slower. A query outline also failed to force the use of a match over cross strategy.

```
select s.subject_code, s.date_audit, s.audit_username
  from subjects_d_audit s
  where s.date_audit =
        (select max(s1.date_audit) from subjects_d_audit s1
         where s1.subject_code = s.subject_code
         )
  and exists
        (select s2.subject_code from subjects s2
         where s2.subject_code = s.subject_code
         )
;
~S: Outline "QO_A115A0E044FFD4BF_00000000" used
~S: Full compliance with the outline was not possible
%RDMS-F-OUTLINE_FAILED, could not comply with mandatory query outline directives
```

Here is the modified outline.

```
create outline QO_A115A0E044FFD4BF_00000000
id 'A115A0E044FFD4BF3957A95575671E8C'
mode 0
as (
  query (
-- For loop
    subquery (
      SUBJECTS_D_AUDIT 0      access path sequential
    !   join by cross to
      join by match to
    subquery (
      SUBJECTS_D_AUDIT 1      access path sequential
    )
    join by cross to
    subquery (
      SUBJECTS 2      access path index      SUBJECTS_PKEY
    )
  )
)
)
compliance mandatory ;
```

The key parts of this query which contributed to the situation leading to the error are these:

1. the main select query with 2 or more subquery's in the where clause
2. each subquery is joined to the common column of the main context

Oracle Rdb7 Release 7.0.1.5 introduced a problem when a fix was made for Problem Report 771079.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.2 SQL Errors Fixed

4.2.1 Unexpected UNSDATASS Error Reported by SQL Precompiler and Module Language

Bugs 951824 and 1033571

The DATE VMS data type included from the Oracle CDD/Repository was not correctly handled by the CAST function within the SQL precompiler and module language compilers. This resulted in the following error.

```
$ sql$pre/cobol/copy_dict /list/copy_list test.sco
                                WHERE COL2 = CAST( :F_DATE_VMS AS DATE ANSI )
                                                1
%SQL-F-UNSDATASS, (1) Unsupported date/time assignment from
F_DATE_VMS to <cast type>
```

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.2.2 SQL IMPORT No Longer Evaluates Table and Column Constraints

In prior versions of Oracle Rdb, the SQL IMPORT statement would validate each constraint as it was applied to the recreated tables in the database. This could be a time consuming step during IMPORT requiring multiple scans of the source table.

This problem has been corrected in Oracle Rdb7 Release 7.0.4. The SQL IMPORT statement no longer requires that constraints be validated. Eliminating this step should reduce the time taken to IMPORT a database containing many constraints. Oracle recommends using RMU/VERIFY/CONSTRAINTS to check constraints.

Note

Users of the RDO IMPORT command are encouraged to use the SQL IMPORT to benefit from this change in behavior.

4.2.3 Unexpected INVACC_OUT_PARA Error Generated by CREATE MODULE

In previous versions of Oracle Rdb7, the CALL statement in a stored procedure or function might cause CREATE MODULE to fail unexpectedly.

The following example shows the error which may be generated by the CREATE MODULE statement.

```

SQL> create module SAMPLE_MODULE_P
cont>     language SQL
cont>
cont>     procedure P1 (out :a integer);
cont>     set :a = 0;
cont>
cont> end module;
SQL>
SQL> create module SAMPLE_MODULE_Q
cont>     language SQL
cont>
cont>     procedure Q1 (out :c integer);
cont>     call P1 (:c);
cont>
cont> end module;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-INVALID_BLR, request BLR is incorrect at offset 58
-RDMS-E-INVACC_OUT_PARA, attempt to read from an OUT parameter

```

This error is generated when a parameter declared as OUT is passed to a stored procedure that similarly expects an OUT parameter. Oracle Rdb was incorrectly requiring IN access to the parameter.

As a workaround, the parameter may be declared as INOUT to avoid this error.

This problem has been corrected in Oracle Rdb7 Release 7.0.4. The Oracle Rdb Server now correctly checks the parameter mode and no longer requires the parameter to be declared as INOUT in this case.

4.2.4 Changed Behavior for CAST of Date/Time Values With Seconds Field

Bug 1075663

On most VAX and Alpha AXP hardware, the VMS system time is maintained to at least 1 millisecond intervals, which is more precise than is currently supported by Oracle Rdb.

Applications which accept the date/time using system services (SYSS\$GETTIM, SYSS\$BINTIM, etc) and insert those values using SQL must be aware that these date/time values will be truncated to 100th of a second by formatting routines such as SYSS\$ASCTIM, LIB\$FORMAT_DATE_TIME and the SQL statements SELECT, PRINT, etc.

When the displayed results are subsequently used as input to queries it is possible that no matches will be found because the values do not include the full fractional seconds precision. The following example shows the potential problem.

```

SQL> select ts_col from ts_table;
TS_COL
10-DEC-1999 10:27:10.80
10-DEC-1999 10:27:11.21
10-DEC-1999 10:27:11.21
10-DEC-1999 10:27:11.21
10-DEC-1999 10:27:11.21
10-DEC-1999 10:27:11.21
10-DEC-1999 10:27:11.22
10-DEC-1999 10:27:11.22
10-DEC-1999 10:27:11.22
9 rows selected
SQL> select * from ts_table
cont> where ts_col = date vms'10-DEC-1999 10:27:10.80';
0 rows selected
SQL>

```

Oracle Rdb currently only supports times and timestamps up to 100ths of a second precision, e.g. `TIMESTAMP(2)`. Starting with Rdb Version 5.1, all Rdb Server generated timestamps (`CURRENT_TIME` and `CURRENT_TIMESTAMP`) are automatically truncated to 100ths of a second. Oracle therefore recommends that these functions be used in preference to the OpenVMS system services to avoid this problem.

However, if timestamp values must be derived from an external source then care must be taken to query or store those values with correct truncation of the fractional seconds precision.

Displaying full precision of seconds

The run-time library routine `LIB$FORMAT_DATE_TIME` can be used to format the higher precision seconds fields. This routine is used by interactive SQL for `DATE VMS` types. First a date formatting logical names must be defined which includes the higher precision, as in the following example.

```
$ DEFINE/EXEC/TABLE=LNM$DT_FORMAT_TABLE -
  LIB$TIME_FORMAT_502 "!H02:!M0:!S0.!C7"
```

Once this logical is defined it can be used by any application which formats using `LIB$FORMAT_DATE_TIME`.

```
SQL> set date format date 1, time 502
SQL> select ts_col from ts_table;
TS_COL
10-DEC-1999 10:27:10.8064904
10-DEC-1999 10:27:11.2175969
10-DEC-1999 10:27:11.2185734
10-DEC-1999 10:27:11.2195499
10-DEC-1999 10:27:11.2195499
10-DEC-1999 10:27:11.2195499
10-DEC-1999 10:27:11.2205264
10-DEC-1999 10:27:11.2205264
10-DEC-1999 10:27:11.2205264
9 rows selected
SQL> select * from ts_table
cont> where ts_col between date vms'10-DEC-1999 10:27:10.80'
cont> and date vms'10-DEC-1999 10:27:10.81';
TS_COL
10-DEC-1999 10:27:10.8064904
1 row selected
SQL>
```

Oracle Rdb7 Release 7.0.4 has been enhanced so that the `CAST` operator now efficiently truncates these extra fractional seconds precision when you use the same input and output data types. That is, if you cast a `DATE VMS` type to `DATE VMS`, the fractional seconds precision is enforced. The same is true for `TIME`, `TIMESTAMP` and `INTERVAL` types which include the `SECOND` field.

The following example shows the query result after truncation using `CAST` function.

```

SQL> select cast(ts_col as date vms) from ts_table;
10-DEC-1999 10:27:10.8000000
10-DEC-1999 10:27:11.2100000
10-DEC-1999 10:27:11.2100000
10-DEC-1999 10:27:11.2100000
10-DEC-1999 10:27:11.2100000
10-DEC-1999 10:27:11.2100000
10-DEC-1999 10:27:11.2200000
10-DEC-1999 10:27:11.2200000
10-DEC-1999 10:27:11.2200000
9 rows selected
SQL> select * from ts_table
cont> where cast(ts_col as date vms) = date vms'10-DEC-1999 10:27:10.80';
TS_COL
10-DEC-1999 10:27:10.8064904
1 row selected
SQL>

```

4.2.5 SQL Rejects Queries Which Use Column Named VALUE

Bug 1149113

In prior versions of Oracle Rdb, using a column named VALUE was prohibited because of the special nature of this keyword. VALUE is a special identifier reserved for use in a domain CHECK constraint definition. Attempts to use such a column caused a fatal error for DML statements (INSERT, SELECT, DELETE and UPDATE) as shown in this simple example.

```

SQL> select value from v;
%SQL-F-VALUEILL, VALUE cannot be used outside of a domain constraint

```

While it is true that VALUE is a reserved word in the ANSI and ISO SQL Standards, other similar keywords cause an information message to be generated so that older applications can continue to execute unchanged. However, this VALUEILL error prevented applications from working with more recent versions of Oracle Rdb.

With this release of Oracle Rdb, the VALUEILL error is no longer reported and VALUE is treated in the same way as other reserved words. That is, a warning is issued by default. The query will fail if a dialect is established such as SQL92.

```

SQL> select value from v;
%SQL-I-DEPR_FEATURE, Deprecated Feature: Keyword VALUE used as an identifier
0 rows selected
SQL> set dialect 'sql92';
SQL> select value from v;
%SQL-F-RES_WORD_AS_IDE, Keyword VALUE used as an identifier

```

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.3 Oracle RMU Errors Fixed

4.3.1 RMU Extract Has Enhanced Extract of Conditional Expressions

Oracle Rdb7 Release 7.0.4 improves the extraction of the conditional expressions COALESCE, NVL, NULLIF, and simple CASE expressions.

In prior releases, these expressions were incorrectly extracted, and may have appeared as searched CASE expressions. This occurred because the pattern matching algorithm often didn't find a match for these expressions. This release enhances the pattern matching to match correctly these expressions.

The side effect of these changes is that some searched CASE expressions may be extracted as an alternate and more compact form of the conditional expression.

The following list shows the equivalent expressions matched by RMU Extract.

- NULLIF (a, b) is equivalent to

```
CASE
  WHEN a = b THEN NULL
  ELSE a
END
```

- NVL (a, ..., b) or COALESCE (a, ..., b) is equivalent to

```
CASE
  WHEN a IS NOT NULL THEN a
  ...
  ELSE b
END
```

- The simple CASE expression

```
CASE a
  WHEN b THEN v1
  WHEN NULL THEN v2
  ...
  ELSE v3
END
```

is equivalent to

```
CASE
  WHEN a = b THEN v1
  WHEN a IS NULL THEN v2
  ...
  ELSE v3
END
```

RMU Extract tries to decode the internal representation to as compact a SQL expression as possible.

4.3.2 RMU/REPLICATE AFTER START Command Fails on TCP/IP With Large Port Numbers

When the TCP/IP service for Hot Standby is defined with a port number larger than 32,767, the network connection would fail due to incorrect network port to host port translation of the port number.

```
$ rmu/replicate after_journal start m_testdb.rdb -
  /standby=node::device-directory:s_testdb.rdb
%COSI-F-CONNCFAIL, connect over network timed-out or failed
```

This problem has been corrected in Oracle Rdb7 Release 7.0.4. With this release, TCP/IP port numbers up to 65,535 will be supported.

4.3.3 SHOW STATS Cannot Replay /OPTIONS=ROW_CACHE Input File

The RMU Show Statistic utility was unable to replay binary output files created with the /OPTIONS=ROW_CACHE or /OPTIONS=ALL qualifiers. The problem only occurs when the database has row caching enabled.

The only work-around is to *not* use the /OPTIONS=ROW_CACHE qualifier.

This problem has been corrected in Oracle Rdb7 Release 7.0.4. The /OPTIONS=ROW_CACHE and /OPTIONS=ALL qualifiers now work correctly.

4.3.4 RMU/SHOW LOCKS Difficult to Identify Lock Conflict Culprit

Each line of the RMU/SHOW LOCKS utility output shows the process that is waiting and the process that is blocking it. At least one of the blocking processes is not in the list of waiting processes. In other words, the process is either running a long transaction or, more likely, it's waiting for a non-database event. If this process is terminated or forced to finish its transaction, the waiting processes start to move, and frequently the blocks all clear.

```

Waiting      Blocker
0001         0002
0002         0003
0003         0004  <-- stop/id=0004 may well free things up
0005         0003
0006         0005  <-- stop/id=0005 would only help 0006

```

Maybe processes 0001-0006 are all culprits, but there is a sense in which process 0004 is more culpable.

There is no workaround to this problem other than manually searching the RMU/SHOW LOCKS /MODE=WAITING output manually.

This problem has been corrected in Oracle Rdb7 Release 7.0.4. The RMU/SHOW LOCKS utility has been enhanced to support a new display mode, /MODE=CULPRIT.

The /MODE=CULPRIT output is a sanitized version of the /MODE=WAITING output. The /MODE=CULPRIT qualifier displays only the set of locks for processes that are blocking other processes but are themselves not blocked. This output represents the processes that are the source of database stalls and performance degradation.

In the following real-world example, one process is blocking the entire application. Compare the difference in the output between the /MODE=WAITING and /MODE=CULPRIT output.

The /MODE=WAITING qualifier displays the following output:

```

=====
SHOW LOCKS/LOCK/MODE=WAITING Information
=====
-----
Resource: record 109:1085:0
-----
          ProcessID Process Name          Lock ID   System ID Requested  Granted
-----
Blocker: 3C806080  RICK10.....  23002C6A  000100E4  PR           NL
Waiting: 3C806081  RICK8.....   33004A4A  000100E4  PR           NL
Waiting: 3C805C86  RICK14.....  470088DA  000100E4  PR           NL
Waiting: 3C805E7E  RICK4.....   410002FA  000100E4  PR           NL
-----
Resource: record 109:1085:0
-----
          ProcessID Process Name          Lock ID   System ID Requested  Granted
-----
Blocker: 3C806081  RICK8.....   33004A4A  000100E4  PR           NL
Waiting: 3C805C86  RICK14.....  470088DA  000100E4  PR           NL
Waiting: 3C805E7E  RICK4.....   410002FA  000100E4  PR           NL
-----
Resource: record 109:1085:0

```


	ProcessID	Process Name	Lock ID	System ID	Requested	Granted
Blocker:	3C806083	RICK12.....	0E008171	000100E4	PR	PR
Waiting:	3C805E82	RICK9.....	2A0087FF	000100E4	EX	PR
Waiting:	3C805A84	RICK11.....	3B00F074	000100E4	PR	NL
Waiting:	3C806085	RICK13.....	3200EF7E	000100E4	PR	NL
Waiting:	3C806080	RICK10.....	23002C6A	000100E4	PR	NL
Waiting:	3C806081	RICK8.....	33004A4A	000100E4	PR	NL
Waiting:	3C805C86	RICK14.....	470088DA	000100E4	PR	NL
Waiting:	3C805E7E	RICK4.....	410002FA	000100E4	PR	NL

Resource: record 109:1085:0

	ProcessID	Process Name	Lock ID	System ID	Requested	Granted
Blocker:	3C805E82	RICK9.....	2A0087FF	000100E4	EX	PR
Waiting:	3C805A84	RICK11.....	3B00F074	000100E4	PR	NL
Waiting:	3C806085	RICK13.....	3200EF7E	000100E4	PR	NL
Waiting:	3C806080	RICK10.....	23002C6A	000100E4	PR	NL
Waiting:	3C806081	RICK8.....	33004A4A	000100E4	PR	NL
Waiting:	3C805C86	RICK14.....	470088DA	000100E4	PR	NL
Waiting:	3C805E7E	RICK4.....	410002FA	000100E4	PR	NL

Resource: record 109:1085:0

	ProcessID	Process Name	Lock ID	System ID	Requested	Granted
Blocker:	3C8004DC	RICK5.....	370088C2	000100E4	PR	PR
Waiting:	3C805E82	RICK9.....	2A0087FF	000100E4	EX	PR
Waiting:	3C805A84	RICK11.....	3B00F074	000100E4	PR	NL
Waiting:	3C806085	RICK13.....	3200EF7E	000100E4	PR	NL
Waiting:	3C806080	RICK10.....	23002C6A	000100E4	PR	NL
Waiting:	3C806081	RICK8.....	33004A4A	000100E4	PR	NL
Waiting:	3C805C86	RICK14.....	470088DA	000100E4	PR	NL
Waiting:	3C805E7E	RICK4.....	410002FA	000100E4	PR	NL

Resource: record 109:1085:0

	ProcessID	Process Name	Lock ID	System ID	Requested	Granted
Blocker:	3C805C86	RICK14.....	470088DA	000100E4	PR	NL
Waiting:	3C805E7E	RICK4.....	410002FA	000100E4	PR	NL

Resource: record 109:1660:1

	ProcessID	Process Name	Lock ID	System ID	Requested	Granted
Blocker:	3C8004DC	RICK5.....	2D00D032	000100E4	EX	EX
Waiting:	3C806083	RICK12.....	5700D6F9	000100E4	EX	NL

Resource: record 109:1085:0

	ProcessID	Process Name	Lock ID	System ID	Requested	Granted
Blocker:	3C806085	RICK13.....	3200EF7E	000100E4	PR	NL
Waiting:	3C806080	RICK10.....	23002C6A	000100E4	PR	NL
Waiting:	3C806081	RICK8.....	33004A4A	000100E4	PR	NL
Waiting:	3C805C86	RICK14.....	470088DA	000100E4	PR	NL
Waiting:	3C805E7E	RICK4.....	410002FA	000100E4	PR	NL

Resource: record 109:1085:0

	ProcessID	Process Name	Lock ID	System ID	Requested	Granted
Blocker:	3C805A84	RICK11.....	3B00F074	000100E4	PR	NL
Waiting:	3C806085	RICK13.....	3200EF7E	000100E4	PR	NL
Waiting:	3C806080	RICK10.....	23002C6A	000100E4	PR	NL
Waiting:	3C806081	RICK8.....	33004A4A	000100E4	PR	NL
Waiting:	3C805C86	RICK14.....	470088DA	000100E4	PR	NL
Waiting:	3C805E7E	RICK4.....	410002FA	000100E4	PR	NL

The /MODE=CULPRIT qualifier displays the following output:

```

=====
SHOW LOCKS/LOCK/MODE=CULPRIT Information
=====
-----
Resource: record 109:1085:0
ProcessID Process Name      Lock ID   System ID Requested  Granted
-----
Blocker:  3C8004DC  RICK5.....  370088C2  000100E4  PR         PR
Waiting:  3C805E82  RICK9.....  2A0087FF  000100E4  EX         PR
Waiting:  3C805A84  RICK11..... 3B00F074  000100E4  PR         NL
Waiting:  3C806085  RICK13..... 3200EF7E  000100E4  PR         NL
Waiting:  3C806080  RICK10..... 23002C6A  000100E4  PR         NL
Waiting:  3C806081  RICK8.....  33004A4A  000100E4  PR         NL
Waiting:  3C805C86  RICK14..... 470088DA  000100E4  PR         NL
Waiting:  3C805E7E  RICK4.....  410002FA  000100E4  PR         NL
-----
Resource: record 109:1660:1
ProcessID Process Name      Lock ID   System ID Requested  Granted
-----
Blocker:  3C8004DC  RICK5.....  2D00D032  000100E4  EX         EX
Waiting:  3C806083  RICK12..... 5700D6F9  000100E4  EX         NL

```

In this example, process 3C8004DC is the culprit of two separate, but probably related, stalls.

4.3.5 RMU BACKUP to Tape Hung if Bad Checksum

Bug 1059787

When a database page contained an invalid checksum, RMU/BACKUP/ONLINE to a tape device hung instead of reporting the error if checksum checking was enabled.

The following example shows a sample RMU BACKUP command line which caused the hang if there was a bad checksum on a database page.

```
RMU/BACKUP/ONLINE/LABEL=BACK01 database.rdb TAPE:database.rbf
```

The following shows the corrected behavior: an error message is output and the backup to tape reports the fatal error and does not hang.

```

RMU/BACKUP/ONLINE/LABEL=BACK01 database.rdb TAPE:database.rbf
%RMU-F-CANTREADDBS, error reading pages 2:3-3
-RMU-F-CHECKSUM, checksum error - computed 67C3D4E8, page contained 00003039
%RMU-F-FATALERR, fatal error on BACKUP

```

As a workaround, to avoid the problem do not enable checksum checking for RMU BACKUP to tape.

```
RMU/BACKUP/NOCHECKSUM/ONLINE/LABEL=BACK01 database.rdb TAPE:database.rbf
```

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.3.6 RMU BACKUP to Tape Hung on QUIT Response to Wrong Label Message

RMU BACKUP to tape devices hung when the user chose the "QUIT" response as the reply to the message output by RMU BACKUP when a label was specified in the RMU BACKUP command which did not match the label on the tape device being used for the backup.

The following example shows an RMU BACKUP command line and QUIT response to the wrong label message output by RMU BACKUP which caused RMU BACKUP to tape to hang.

```
RMU/BACKUP/REWIND/LABEL=(badlab01,badlab02)/LOADER MF_PERSONNEL.RDB -
$111$MUA30:MF_PERSONNEL.RBF/MASTER, $111$MUA31:/MASTER
%RMU-I-WRNGLBL, Tape on _$111$MUA30 was incorrectly labeled. Expected GOODLAB
- Found BADLAB01
%RMU-I-TAPEDISPW, Specify tape disposition for _$111$MUA30 (QUIT,INITIALIZE,
RETRY,UNLOAD)
quit
```

The workaround for this problem is to choose an option other than "QUIT" in response to the bad label message or to reenter the RMU BACKUP command specifying a label that matches the label on the tape device.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.3.7 RMU/REPAIR/INIT=FREE_PAGES/ABM Did Not Return an Error

Bug 968268

The RMU/REPAIR documented restriction that the qualifiers /INITIALIZE=FREE_PAGES and /ABM were conflicting qualifiers and could not be used together on the same RMU/REPAIR command line was not enforced by a conflicting qualifiers error message but was allowed.

The following example shows that the /INITIALIZE=FREE_PAGES and /ABM qualifiers were accepted by the RMU/REPAIR command when a conflicting qualifiers error should have been returned.

```
$RMU/REPAIR/ABM/SPAM/INITIALIZE=FREE_PAGES/AREA=AREA_NAME MF_PERSONNEL
```

The following example shows that an error is now returned and the command is not accepted.

```
$RMU/REPAIR/ABM/SPAM/INITIALIZE=FREE_PAGES/AREA=AREA_NAME MF_PERSONNEL
%RMU-F-CONFLSWIT, conflicting qualifiers /ABM and /INITIALIZE=FREE_PAGES
```

As a workaround, do not include the /ABM and /INITIALIZE=FREE_PAGES qualifiers in the same RMU/REPAIR command line.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.3.8 Incorrect BADIDXREL Messages From Online RMU Verify

Bugs 883349 and 1039089

An online RMU VERIFY of a database index where /TRANSACTION_TYPE=READ_ONLY was specified sometimes output incorrect RMU-W-BADIDXREL warning messages when the index was being concurrently modified by other users. These same BADIDXREL messages were not output if the index was not being modified during the online verify or if READ_ONLY was not specified with the /TRANSACTION_TYPE qualifier.

The following example shows the RMU VERIFY command for verifying a database index using the /TRANSACTION_TYPE=READ_ONLY qualifier and the resulting RMU-W-BADIDXREL warning message which was not output if /TRANSACTION_TYPE=READ_ONLY was not specified.

```
$ rmu/verify/noroot/transaction_type=read_only -  
/index=(db_index)/data rdb_database  
%RMU-W-BADIDXREL, Index DB_INDEX either points to a non-existent record or  
has multiple pointers to a record in table RDB_TABLE.  
The logical dbkey in the index is 527:2324:1.
```

The workaround for this problem is to use the /TRANSACTION_TYPE=READ_ONLY qualifier when no user transaction is modifying the database index being verified or to specify another /TRANSACTION_TYPE such as PROTECTED (the default) or EXCLUSIVE.

```
$ rmu/verify/noroot/transaction_type=exclusive -  
/index=(db_index)/data rdb_database
```

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.3.9 RMU VERIFY Did Not Find a .RDA File After an RMU MOVE

RMU/VERIFY did not find a .RDA database area file which had been updated to a new version by the RMU/MOVE of the associated database snapshot file which had been executed on another node of the cluster.

The following example shows the error.

```
On Node1:  
$ RMU/OPEN/ACC=UNR MF_PERSONNEL  
On Node2:  
$ RMU/OPEN/ACC=UNR MF_PERSONNEL  
$ CREATE [.TEST] /DIR  
$ RMU/MOVE/ONL/LOG MF_PERSONNEL RESUMES /SNAP=FILE=[.TEST]  
On Node1:  
$ RMU/VERIFY/LOG/TRANS=READ_ONLY/AREA=RESUMES/SNAP MF_PERSONNEL  
%RMU-I-BGNROOVER, beginning root verification  
%RMU-F-OPNFILERR, error opening file  
DISK:[DIRECTORY]RESUMES.RDA;1  
%RMU-F-FILNOTFND, file not found  
%RMU-E-BDAREAOPN, unable to open file  
DISK:[DIRECTORY]RESUMES.RDA;1 for storage area RESUMES  
%RMU-F-ABORTVER, fatal error encountered; aborting verification
```

A workaround for this problem is to do the RMU/MOVE on the same node as the RMU/VERIFY.

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

4.4 Row Cache Errors Fixed

4.4.1 Row Cache Server Operator Notification

Similar to other Oracle Rdb7 database servers, the Row Cache Server (RCS) process now sends start and terminate messages to the system operator if database operator notifications are enabled.

The following example shows the format of the Row Cache Server operator message:

```

$ REPLY/ENABLE=CENTRAL
%%%%%%%%%% OPCOM 28-SEP-1999 17:16:57.32 %%%%%%%%%%%
Operator TTA0: has been enabled, username RC

%%%%%%%%%% OPCOM 28-SEP-1999 17:16:57.33 %%%%%%%%%%%
Operator status for operator TTA0:
CENTRAL

$ RMU/OPEN DUA0:[DB]MFP
%%%%%%%%%% OPCOM 28-SEP-1999 17:15:47.66 %%%%%%%%%%%
Message from user RDBVMS on RYEROX
Oracle Rdb X7.1-00 Database DUA0:[DB]MFP.RDB;1 Event Notification
Row Cache Server started

```

4.4.2 Row Cache Did Not Avoid Certain Database Writes

In certain situations, the Oracle Rdb7 Row Cache feature did not avoid database update I/O that it otherwise could have avoided. In particular, when a database record was modified when it was not originally in the cache, it is possible that the database page containing the row could be written back to the database where it otherwise would not have to be.

Some applications may find a performance improvement when using the Row Cache feature with Oracle Rdb7 Release 7.0.4 due to the reduction in unneeded database write I/O for some update operations.

4.4.3 RMU /CLOSE /WAIT Would Not Always Wait When Row Cache Enabled

When using the Row Cache feature with many or large caches, it was possible that the RMU /CLOSE /WAIT command could return to the user with the database still actually being shut down.

This problem has been corrected in Oracle Rdb7 Release 7.0.4. The RMU /CLOSE /WAIT command now does an additional check to make sure that the database is not open before returning to the user.

4.5 Hot Standby Errors Fixed

4.5.1 RMU/REPLICATE AFTER START Command Fails Due to Lost AIJ Write

There is a situation where Hot Standby fails but has already committed a transaction that did not get written to the AIJ journal. During re-start of Hot Standby, the lost write is before the last committed transaction causing re-start to fail. The following error message is returned during restart.

```

$ rmu/replicate after_journal start s_testdb.rdb -
  /master_root=m_testdb.rdb
%RDMS-F-CANTSTARTLRS, error starting AIJ Log Roll-Forward Server process
%RMU-F-FATALRDB, Fatal error while accessing Oracle Rdb.

```

The following messages would appear in the LRS log file:

```

23-OCT-1999 07:49:41 - Transaction recovery not started during restart
23-OCT-1999 07:49:41 - This usually occurs when a manual roll-forward operation
23-OCT-1999 07:49:41 - using master database AIJ journals did not fully complete
23-OCT-1999 07:49:41 - This is sometimes caused by an AIJ switch-over operation
23-OCT-1999 07:49:41 - while Hot Standby is inactive

23-OCT-1999 07:49:41 - Failure reason: %RDMS-F-CANTSTARTLRS,
                        error starting AIJ Log Roll-Forward Server process

```

This problem has been corrected in Oracle Rdb7 Release 7.0.4.

Documentation Corrections

This chapter provides information not currently available in the Oracle Rdb7 documentation set.

5.1 Documentation Corrections

5.1.1 RMU /UNLOAD /AFTER_JOURNAL NULL Bit Vector Clarification

Each output record from the RMU /UNLOAD /AFTER_JOURNAL command includes a vector (array) of bits. There is one bit for each field in the data record. If a null bit value is 1, the corresponding field is NULL; if a null bit value is 0, the corresponding field is not NULL and contains an actual data value. The contents of a data field that is NULL are not initialized and are not predictable.

The null bit vector begins on a byte boundary. The field RDB\$LM_NBV_LEN indicates the number of valid bits (and thus, the number of columns in the table). Any extra bits in the final byte of the vector after the final null bit are unused and the contents are unpredictable.

The following example C program demonstrates one possible way of reading and parsing a binary output file (including the null bit vector) from the RMU /UNLOAD /AFTER_JOURNAL command. This sample program has been tested using Oracle Rdb V7.0.5 and Compaq C V6.2-009 on OpenVMS Alpha V7.2-1. It is meant to be used as a template for writing your own program.

```

/* DATATYPES.C */

#include <stdio.h>
#include <descrip.h>
#include <starlet.h>
#include <string.h>

#pragma member_alignment __save
#pragma nomember_alignment

struct { /* Database key structure */
    unsigned short    lno;    /* line number */
    unsigned int      pno;    /* page number */
    unsigned short    dbid;   /* area number */
} dbkey;

typedef struct { /* Null bit vector with one bit for each column */
    unsigned          n_tinyint    :1;
    unsigned          n_smallint   :1;
    unsigned          n_integer    :1;
    unsigned          n_bigint     :1;
    unsigned          n_double     :1;
    unsigned          n_real       :1;
    unsigned          n_fixstr     :1;
    unsigned          n_varstr     :1;
} nbv_t;

```

```

struct { /* LogMiner output record structure for table DATATYPES */
    char          rdb$lm_action;
    char          rdb$lm_relation_name [31];
    int           rdb$lm_record_type;
    short         rdb$lm_data_len;
    short         rdb$lm_nbv_len;
    __int64       rdb$lm_dbk;
    __int64       rdb$lm_start_tad;
    __int64       rdb$lm_commit_tad;
    __int64       rdb$lm_tsn;
    short         rdb$lm_record_version;
    char          f_tinyint;
    short         f_smallint;
    int           f_integer;
    __int64       f_bigint;
    double        f_double;
    float         f_real;
    char          f_fixstr[10];
    short         f_varstr_len; /* length of varchar */
    char          f_varstr[10]; /* data of varchar */
    nbv_t         nbv;
} lm;

#pragma member_alignment __restore

main ()
{
    char timbuf [24];
    struct dsc$descriptor_s dsc = {
        23, DSC$K_DTYPE_T, DSC$K_CLASS_S, timbuf};
    FILE *fp = fopen ("datatypes.dat", "r", "ctx=bin");

    memset (&timbuf, 0, sizeof(timbuf));

    while (fread (&lm, sizeof(lm), 1, fp) != 0)
    {
        printf ("Action      = %c\n",      lm.rdb$lm_action);
        printf ("Table        = %.*s\n",      sizeof(lm.rdb$lm_relation_name),
            lm.rdb$lm_relation_name);

        printf ("Type          = %d\n",      lm.rdb$lm_record_type);
        printf ("Data Len     = %d\n",      lm.rdb$lm_data_len);
        printf ("Null Bits    = %d\n",      lm.rdb$lm_nbv_len);

        memcpy (&dbkey, &lm.rdb$lm_dbk, sizeof(lm.rdb$lm_dbk));
        printf ("DBKEY        = %d:%d:%d\n", dbkey.dbid,
            dbkey.pno,
            dbkey.lno);

        sys$asctim (0, &dsc, &lm.rdb$lm_start_tad, 0);
        printf ("Start TAD    = %s\n", timbuf);

        sys$asctim (0, &dsc, &lm.rdb$lm_commit_tad, 0);
        printf ("Commit TAD   = %s\n", timbuf);

        printf ("TSN          = %ld\n",      lm.rdb$lm_tsn);
        printf ("Version      = %d\n",      lm.rdb$lm_record_version);

        if (lm.nbv.n_tinyint == 0)
            printf ("f_tinyint    = %d\n",      lm.f_tinyint);
        else
            printf ("f_tinyint    = NULL\n");

        if (lm.nbv.n_smallint == 0)
            printf ("f_smallint   = %d\n",      lm.f_smallint);
        else
            printf ("f_smallint   = NULL\n");

        if (lm.nbv.n_integer == 0)
            printf ("f_integer    = %d\n",      lm.f_integer);
        else
            printf ("f_integer    = NULL\n");
    }
}

```



```

        if (lm.nbv.n_bigint == 0)
            printf ("f_bigint    = %Ld\n", lm.f_bigint);
        else
            printf ("f_bigint    = NULL\n");
        if (lm.nbv.n_double == 0)
            printf ("f_double   = %f\n", lm.f_double);
        else
            printf ("f_double   = NULL\n");
        if (lm.nbv.n_real == 0)
            printf ("f_real     = %f\n", lm.f_real);
        else
            printf ("f_real     = NULL\n");
        if (lm.nbv.n_fixstr == 0)
            printf ("f_fixstr   = %.*s\n", sizeof (lm.f_fixstr),
                    lm.f_fixstr);
        else
            printf ("f_fixstr   = NULL\n");
        if (lm.nbv.n_varstr == 0)
            printf ("f_varstr   = %.*s\n", lm.f_varstr_len, lm.f_varstr);
        else
            printf ("f_varstr   = NULL\n");
        printf ("\n");
    }
}

```

Example sequence of commands to create a table, unload the data and display the contents with this program:

```

SQL> ATTACH 'FILE MF_PERSONNEL';
SQL> CREATE TABLE DATATYPES (
    F_TINYINT TINYINT
    ,F_SMALLINT SMALLINT
    ,F_INTEGER INTEGER
    ,F_BIGINT BIGINT
    ,F_DOUBLE DOUBLE PRECISION
    ,F_REAL REAL
    ,F_FIXSTR CHAR (10)
    ,F_VARSTR VARCHAR (10));
SQL> COMMIT;
SQL> INSERT INTO DATATYPES VALUES (1, NULL, 2, NULL, 3, NULL, 'THIS', NULL);
SQL> INSERT INTO DATATYPES VALUES (NULL, 4, NULL, 5, NULL, 6, NULL, 'THAT');
SQL> COMMIT;
SQL> EXIT;
$ RMU /BACKUP /AFTER_JOURNAL MF_PERSONNEL AIJBCK.AIJ
$ RMU /UNLOAD /AFTER_JOURNAL MF_PERSONNEL AIJBCK.AIJ -
  /TABLE = (NAME=DATATYPES, OUTPUT=DATATYPES.DAT)
$ CC DATATYPES.C
$ LINK DATATYPES.OBJ
$ RUN DATATYPES.EXE

```

5.1.2 Location of Host Source File Generated by the SQL Precompilers

Bug 478898

When the SQL precompiler generates host source files (like .c, .pas, .for) from the precompiler source files, it locates these files based on the /obj qualifier located on the command line given to the SQL precompiler.

The following examples show the location where the host source file is generated. When /obj is not specified on the command line, the object and the host source file take the name of the SQL precompiler with the extensions of .obj and .c respectively.

```

LUND> sqlpre/cc scc_try_mli_successful.sc
LUND> dir scc_try_mli_successful.*

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.C;1          SCC_TRY_MLI_SUCCESSFUL.OBJ;2
SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 3 files.

```

When /obj is specified on the command line, the object and the host source take the name given on the qualifier switch. It uses the default of the SQL precompiler source if a filespec is not specified. It uses the defaults of .obj and .c if the extension is not specified. If the host language is other than C, then it uses the appropriate host source extension (like .pas, .for, etc). The files also default to the current directory if a directory spec is not specified.

```

LUND> sqlpre/cc/obj=myobj scc_try_mli_successful.sc
LUND> dir scc_try_mli_successful.*

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 1 file.
LUND> dir myobj.*

Directory MYDISK:[LUND]

MYOBJ.C;1          MYOBJ.OBJ;2

Total of 2 files.

LUND> sqlpre/cc/obj=MYDISK:[lund.tmp] scc_try_mli_successful.sc
LUND> dir scc_try_mli_successful.*

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 1 file.
LUND> dir MYDISK:[lund.tmp]scc_try_mli_successful.*

Directory MYDISK:[LUND.TMP]

SCC_TRY_MLI_SUCCESSFUL.C;1          SCC_TRY_MLI_SUCCESSFUL.OBJ;2

Total of 2 files.

```

This problem has been corrected in Oracle Rdb7 Release 7.0.6.

5.1.3 Suggestion to Increase GH_RSRVPGCNT Removed

The Oracle Rdb7 for OpenVMS Installation and Configuration Guide contains a section titled "Installing Oracle Rdb Images as Resident on OpenVMS Alpha" that includes information about increasing the value of the OpenVMS system parameter GH_RSRVPGCNT when you modify the RMONSTART.COM or SQL\$STARTUP.COM procedures to install Rdb images with the /RESIDENT qualifier.

Note that modifying the parameter GH_RSRVPGCNT is only ever required if the RMONSTART.COM or SQL\$STARTUP.COM procedures have been manually modified to install Rdb images with the /RESIDENT qualifier. Furthermore, if the RMONSTART.COM and SQL\$STARTUP.COM procedures are executed during the system startup procedure (directly from SYSTARTUP_VMS.COM, for example), then there is no need to modify the GH_RSRVPGCNT parameter.

Oracle and Compaq suggest that you do not modify the value of the GH_RSRVPGCNT system parameter unless it is absolutely required. Some versions of OpenVMS on some hardware platforms require that GH_RSRVPGCNT be zero in order to ensure the highest level of system performance.

5.1.4 Clarification of the DDLDONOTMIX Error Message

Bug 454080

The ALTER DATABASE statement performs two classes of functions: changing the database root structures in the .RDB file and modifying the system metadata in the RDB\$SYSTEM storage area. The first class of changes do not require a transaction to be active. However, the second class requires that a transaction be active. Oracle Rdb does not currently support the mixing of these two classes of ALTER DATABASE clauses.

When you mix clauses that fall into both classes, the error message DDLDONOTMIX "the {SQL-syntax} clause can not be used with some ALTER DATABASE clauses" is displayed, and the ALTER DATABASE statement fails.

```
SQL> alter database filename MF_PERSONNEL
cont> dictionary is not used
cont> add storage area JOB_EXTRA filename JOB_EXTRA;
%RDB-F-BAD_DPB_CONTENT, invalid database parameters in the database parameter
block (DPB)
-RDMS-E-DDLDONOTMIX, the "DICTIONARY IS NOT USED" clause can not be used with
some ALTER DATABASE clauses
```

The following clauses may be mixed with each other but may not appear with other clauses such as ADD STORAGE AREA, or ADD CACHE.

- DICTIONARY IS [NOT] REQUIRED
- DICTIONARY IS NOT USED
- MULTISHEMA IS { ON | OFF }
- CARDINALITY COLLECTION IS { ENABLED | DISABLED }
- METADATA CHANGES ARE { ENABLED | DISABLED }
- WORKLOAD COLLECTION IS { ENABLED | DISABLED }

If the DDLDONOTMIX error is displayed, then restructure the ALTER DATABASE into two statements, one for each class of actions.

```
SQL> alter database filename MF_PERSONNEL
cont> dictionary is not used;
SQL> alter database filename MF_PERSONNEL
cont> add storage area JOB_EXTRA filename JOB_EXTRA;
```

5.1.5 Compressed Sorted Index Entry Stored in Incorrect Storage Area

This note was originally included in the Oracle Rdb7 Release 7.0.1.3 and 7.0.2 Release Notes. The logical name documented in the note for those releases was documented incorrectly. Below is a corrected note.

In specific cases, in versions V6.1 and V7.0 of Oracle Rdb, when a partitioned, compressed sorted index was created after the data was inserted into the table, b-tree entries may have been inserted into the wrong storage area.

All of the following criteria must be met in order for the possibility of this problem to occur:

- CREATE INDEX is issued after there are records already in the table on which the index is being created
- index must be partitioned over a single column
- index must have compression enabled
- scale factor must be zero on the columns of the index
- no collating sequences specified on the columns of the index
- no descending indexes
- MAPPING VALUES must not be specified

RMU/DUMP/AREA=xx will show that the b-tree entry was not stored in the expected storage area. However, in versions V6.1 and V7.0 of Oracle Rdb, the rows of the table can still be successfully retrieved.

The following example shows the problem:

```
create database
  filename foo
create storage area Area_1
  filename Area_1
create storage area Area_2
  filename Area2;

create table T1
  (C1 integer);

! insert data into table prior to index creation
insert into T1 values (0);
commit;

! create index with COMPRESSION ENABLED
create index Index_1
  on T1 (C1)
  enable compression
  store using (C1)
  in Area_1 with limit of (0)
  otherwise in Area_2;
COMMIT;
!
! Dump out the page for b-tree in AREA_1, there are 0 bytes stored.
! There should be 5 bytes stored for the b-tree entry.
!
RMU/DUMP/AREA=AREA_1
.
.
.
      .... total B-tree node size: 430
          0030 2003 0240 line 0 (2:5:0) index: set 48
002F FFFFFFFF FFFF 0244 owner 47:-1:-1
          0000 024C 0 bytes of entries <---***** no entry
          8200 024E level 1, full suffix
00000000000000000000000000000000 0250 unused '.....'
.
.
.
!
! Dump out the page for b-tree in AREA_2, there are 5 bytes stored
!
RMU/DUMP/AREA=AREA_2
.
.
```

```

    .... total B-tree node size: 430
    0031 2003 0240 line 0 (3:5:0) index: set 49
002F FFFFFFFF FFFF 0244 owner 47:-1:-1
          000A 024C 10 bytes of entries
          8200 024E level 1, full suffix
          00 05 0250 5 bytes stored, 0 byte prefix <---entry
0100008000 0252 key '.....'
          22B1 10 0257 pointer 47:554:0
  
```

This problem occurs when index compression is enabled. Therefore, a workaround is to create the index with compression disabled (which is the default). Once this update kit is applied, it is recommended that the index be dropped and recreated with compression enabled to rebuild the b-tree.

Note

In prior versions, the rows were successfully retrieved even though the key values were stored in the wrong storage area. This was due to the range query algorithm skipping empty partitions or scanning extra areas.

However, due to an enhancement in the algorithm for range queries on partitioned SORTED indexes in Oracle Rdb7 Release 7.0.2, the rows of the table which are stored in the incorrect storage areas may not be retrieved when using the partitioned index.

The optimized algorithm now only scans the relevant index areas (and no longer skips over empty areas) resulting in only those rows being returned. Therefore, it is recommended that the index be dropped and re-created. For a short term solution, another alternative is to disable the new optimization by defining the logical RDMSSINDEX_PART_CHECK to 0.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3.

5.1.6 Partition Clause is Optional on CREATE STORAGE MAP

Bug 642158

In the *Oracle Rdb7 SQL Reference Manual*, the syntax diagram for the CREATE STORAGE MAP statement incorrectly shows the partition clause as required syntax. The partition clause is not a required clause.

This correction will appear in the next publication of the *Oracle Rdb SQL Reference Manual*.

5.1.7 Oracle Rdb Logical Names

The *Oracle Rdb7 Guide to Database Performance and Tuning* contains a table in Chapter 2 summarizing the Oracle Rdb logical names and configuration parameters. The information in the following table supersedes the entries for the RDM\$BIND_RUJ_ALLOC_BLKCNT and RDM\$BIND_RUJ_EXTEND_BLKCNT logical names.

Logical Name Configuration Parameter	Function
RDM\$BIND_RUJ_ALLOC_BLKCNT	Allows you to override the default value of the .ruj file. The block count value can be defined between 0 and 2 billion with a default of 127.
RDM\$BIND_RUJ_EXTEND_BLKCNT	Allows you to pre-extend the .ruj files for each process using a database. The block count value can be defined between 0 and 65535 with a default of 127.

5.1.8 Waiting for Client Lock Message

The *Oracle Rdb7 Guide to Database Performance and Tuning* contains a section in Chapter 3 that describes the Performance Monitor Stall Messages screen. The section contains a list describing the “Waiting for” messages. The description of the “waiting for client lock” message was missing from the list.

A client lock indicates that an Oracle Rdb metadata lock is in use. The term client indicates that Oracle Rdb is a client of the Oracle Rdb locking services. The metadata locks are used to guarantee memory copies of the metadata (table, index, and column definitions) are consistent with the on-disk versions.

The “waiting for client lock” message means the database user is requesting an incompatible locking mode. For example, when trying to delete a table which is in use, the drop operation requests a PROTECTED WRITE lock on the metadata object (such as a table) which is incompatible with the existing PROTECTED READ lock currently used by others of the table.

These metadata locks consist of three longwords. The lock is displayed in text format first, followed by its hexadecimal representation. The text version masks out nonprintable characters with a period (.).

The leftmost value seen in the hexadecimal output contains the ID of the object. The following ID describes the tables, routines, modules and storage map areas.

- For tables and views, the ID represents the unique value found in the RDB\$RELATION_ID column of the RDB\$RELATIONS system table for the given table.
- For routines, the ID represents the unique value found in the RDB\$ROUTINE_ID column of the RDB\$ROUTINES system table for the given routine.
- For modules, the ID represents the unique value found in the RDB\$MODULE_ID column of the RDB\$MODULES system table for the given module.
- For storage map areas, the ID presents the physical area ID. The “waiting for client lock” message on storage map areas is very rare. This may be raised for databases that have been converted from versions prior to Oracle Rdb 5.1.

The next value displayed signifies the object type. The following table describes objects and their hexadecimal type values:

Table 5–1 Object Type Values

Object	Hexadecimal Value
Tables or views	00000004
Routines	00000006
Modules	00000015
Storage map areas	0000000E

The last value in the hexadecimal output represents the lock type. The value 55 indicates this is a client lock.

The following example shows a “waiting for client” lock message from the Stall Messages screen:

```
Process.ID Since..... Stall.reason..... Lock.ID.  
46001105:2 10:40:46.38 - waiting for client '.....' 000000190000000400000055  
                                                    1         2         3         4
```

The following list describes each part of the client lock:

- 1 indicates nonprintable characters.
- 2 00000019 indicates unique identifier hex value 19 (RDB\$RELATION_ID = 25).
- 3 00000004 indicates object type 4 which is a table.
- 4 00000055 indicates this is a client lock.

To determine the name of the referenced object given the Lock ID the following queries can be used based on the object type:

```
SQL> SELECT RDB$RELATION_NAME FROM RDB$RELATIONS WHERE RDB$RELATION_ID = 25;  
SQL> SELECT RDB$MODULE_NAME FROM RDB$MODULES WHERE RDB$MODULE_ID = 12;  
SQL> SELECT RDB$ROUTINE_NAME FROM RDB$ROUTINES WHERE RDB$ROUTINE_ID = 7;
```

Note

Because the full client lock output is long, it may require more space than is allotted for the Stall.reason column and therefore can be overwritten by the Lock.ID. column output.

For more detailed lock information, perform the following steps:

1. Press the L option from the horizontal menu to display a menu of Lock IDs.
 2. Select the desired Lock ID.
-

5.1.9 Documentation Error in *Oracle Rdb7 Guide to Database Performance and Tuning*

The *Oracle Rdb7 Guide to Database Performance and Tuning, Volume 2* contains an error in section C.7, “Displaying Sort Statistics with the R Flag”.

When describing the output from this debugging flag, bullet 9 states:

Work File Alloc indicates how many work files were used in the sort operation. A zero (0) value indicates that the sort was accomplished completely in memory.

This is incorrect. This statistic should be described as shown:

Work File Alloc indicates how much space (in blocks) was allocated in the work files for this sort operation. A zero (0) value indicates that the sort was accomplished completely in memory.

This error will be corrected in a future release of *Oracle Rdb Guide to Database Performance and Tuning*.

5.1.10 SET FLAGS Option IGNORE_OUTLINE Not Available

Bug 510968

The *Oracle Rdb7 SQL Reference Manual* described the option IGNORE_OUTLINE in Table 7-6 of the SET FLAGS section. However, this keyword was not implemented in Oracle Rdb7.

This has been corrected in this release of Oracle Rdb7. This keyword is now recognized by the SET FLAGS statement. As a workaround the logical name RDMSS\$BIND_OUTLINE_FLAGS "I" can be used to set this attribute.

5.1.11 SET FLAGS Option INTERNALS Not Described

The *Oracle Rdb7 SQL Reference Manual* does not describe the option INTERNALS in Table 7-6 in the SET FLAGS section. This keyword was available in first release of Oracle Rdb7 and is used to enable debug flags output for internal queries such as constraints and triggers. It can be used in conjunction with other options such as STRATEGY, BLR, and EXECUTION. For example, the following flag settings are equivalent to defining the RDMSS\$DEBUG_FLAGS as ISn and shows the strategy used by the trigger's actions on the AFTER DELETE trigger on the EMPLOYEES table.

```
SQL> SET FLAGS 'STRATEGY, INTERNAL, REQUEST_NAME';
SQL> SHOW FLAGS

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
INTERNALS,STRATEGY,PREFIX,REQUEST_NAMES
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID = '00164';
~S: Trigger name EMPLOYEE_ID_CASCADE_DELETE
Get Temporary relation Retrieval by index of relation DEGREES
Index name DEG_EMP_ID [1:1]
~S: Trigger name EMPLOYEE_ID_CASCADE_DELETE
Get Temporary relation Retrieval by index of relation JOB_HISTORY
Index name JOB_HISTORY_HASH [1:1]
~S: Trigger name EMPLOYEE_ID_CASCADE_DELETE
Get Temporary relation Retrieval by index of relation SALARY_HISTORY
Index name SH_EMPLOYEE_ID [1:1]
~S: Trigger name EMPLOYEE_ID_CASCADE_DELETE
Conjunct Get Retrieval by index of relation DEPARTMENTS
Index name DEPARTMENTS_INDEX [0:0]
Temporary relation Get Retrieval by index of relation EMPLOYEES
Index name EMPLOYEES_HASH [1:1] Direct lookup
1 row deleted
```


5.1.12 Documentation for VALIDATE_ROUTINE Keyword for SET FLAGS

The SET FLAGS section of the *Oracle Rdb7 SQL Reference Manual* omitted the description of the VALIDATE_ROUTINE keyword (which can be negated as NOVALIDATE_ROUTINE). This keyword enables the re-validation of an invalidated stored procedure or function. This flag has the same action as the logical RDMSS\$VALIDATE_ROUTINE or the UNIX environment variable SQL_VALIDATE_ROUTINE described in the *Oracle Rdb7 Guide to Database Performance and Tuning*.

This example shows the re-validation of a stored procedure. When the stored routine is successfully prepared (but not executed), the setting of VALIDATE_ROUTINE causes the entry for this routine in the RDB\$ROUTINES system table to be set as valid.

```
SQL> SET TRANSACTION READ WRITE;
SQL> SET FLAGS 'VALIDATE_ROUTINE';
SQL> SET NOEXECUTE;
SQL> CALL ADD_EMPLOYEE ('Smith');
SQL> SET EXECUTE;
SQL> COMMIT;
```

In this example, the use of the SET NOEXECUTE statement in interactive SQL allows the stored routine to be successfully compiled, but it is not executed.

5.1.13 Documentation for Defining the RDBSERVER Logical Name

Bugs 460611 and 563649.

Sections 4.3.7.1 and 4.3.7.2 in the *Oracle Rdb7 for OpenVMS Installation and Configuration Guide* provide the following examples for defining the RDBSERVER logical name:

```
$ DEFINE RDBSERVER SYS$SYSTEM:RDBSERVER70.EXE
and
$ DEFINE RDBSERVER SYS$SYSTEM:RDBSERVER61.EXE
```

These definitions are inconsistent with other command procedures that attempt to reference the RDBSERVERxx.EXE image. The following is one example where the RDBSERVER.COM procedure references SYS\$COMMON:<SYSEXE> and SYS\$COMMON:[SYSEXE], rather than SYS\$SYSTEM:

```
$ if .not. -
    ((f$locate ("SYS$COMMON:<SYSEXE>", rdbserver_image) .ne. log_len) .or. -
    (f$locate ("SYS$COMMON:[SYSEXE]", rdbserver_image) .ne. log_len))
$ then
$   say "'rdbserver_image' is not found in SYS$COMMON:<SYSEXE>"
$   say "RDBSERVER logical is 'rdbserver_image'"
$   exit
$ endif
```

In this case, if the logical name were defined as instructed in the *Oracle Rdb7 for OpenVMS Installation and Configuration Guide*, the image would not be found.

The *Oracle Rdb7 for OpenVMS Installation and Configuration Guide* should define the logical name as follows:

```
DEFINE RDBSERVER SYS$COMMON:<SYSEXE>RDBSERVER70.EXE
and
DEFINE RDBSERVER SYS$COMMON:<SYSEXE>RDBSERVER61.EXE
```

5.1.14 Undocumented SET Commands and Language Options

The following SET statements were omitted from the Oracle Rdb7 documentation.

5.1.14.1 QUIET COMMIT Option

The SET QUIET COMMIT statement (for interactive and dynamic SQL), the module header option QUIET COMMIT, the /QUIET_COMMIT (and /NOQUIET_COMMIT) qualifier for SQL module language, or the /SQLOPTIONS=QUIET_COMMIT (and NOQUIET_COMMIT) option for the SQL language precompiler allows the programmer to control the behavior of the COMMIT and ROLLBACK statements in cases where there is no active transaction.

By default, if there is no active transaction, SQL will raise an error when COMMIT or ROLLBACK is executed. This default is retained for backward compatibility for applications that may wish to detect the situation. If QUIET COMMIT is set to ON, then a COMMIT or ROLLBACK executes successfully when there is no active transaction.

Note

Within a compound statement, the COMMIT and ROLLBACK statements in this case are ignored.

Examples

In interactive or dynamic SQL, the following SET command can be used to disable or enable error reporting for COMMIT and ROLLBACK when no transaction is active. The parameter to the SET command is a string literal or host variable containing the keyword ON or OFF. The keywords may be in any case (upper, lower, or mixed).

```
SQL> COMMIT;
%SQL-F-NO_TXNOUT, No transaction outstanding
SQL> ROLLBACK;
%SQL-F-NO_TXNOUT, No transaction outstanding
SQL> SET QUIET COMMIT 'on';
SQL> ROLLBACK;
SQL> COMMIT;
SQL> SET QUIET COMMIT 'off';
SQL> COMMIT;
%SQL-F-NO_TXNOUT, No transaction outstanding
```

In the SQL module language or precompiler header, the clause QUIET COMMIT can be used to disable or enable error reporting for COMMIT and ROLLBACK when no transaction is active. The keyword ON or OFF must be used to enable or disable this feature. The following example enables QUIET COMMIT so that no error is reported if a COMMIT is executed when no transaction is active. For example:

```
MODULE TXN_CONTROL
LANGUAGE BASIC
PARAMETER COLONS
QUIET COMMIT ON

PROCEDURE S_TXN (SQLCODE);
SET TRANSACTION READ WRITE;

PROCEDURE C_TXN (SQLCODE);
COMMIT;
```

5.1.14.2 COMPOUND TRANSACTIONS Option

The SET COMPOUND TRANSACTIONS statement (for interactive and dynamic SQL) and the module header option COMPOUND TRANSACTIONS allows the programmer to control the SQL behavior for starting default transactions for compound statements.

By default, if there is no current transaction, SQL will start a transaction before executing a compound statement or stored procedure. However, this may conflict with the actions within the procedure, or may start a transaction for no reason if the procedure body does not perform any database access. This default is retained for backward compatibility for applications that may expect a transaction to be started for the procedure.

If COMPOUND TRANSACTIONS is set to EXTERNAL, then SQL starts a transaction before executing the procedure; otherwise, if it is set to INTERNAL, it allows the procedure to start a transaction as required by the procedure execution.

Examples

In interactive or dynamic SQL, the following SET command can be used to disable or enable transactions started by the SQL interface. The parameter to the SET command is a string literal or host variable containing the keyword INTERNAL or EXTERNAL. The keywords may be in any case (upper, lower, or mixed). For example:

```
SQL> SET COMPOUND TRANSACTIONS 'internal';
SQL> CALL START_TXN_AND_COMMIT ();
SQL> SET COMPOUND TRANSACTIONS 'external';
SQL> CALL UPDATE_EMPLOYEES (...);
```

In the SQL module language or precompiler header, the clause COMPOUND TRANSACTIONS can be used to disable or enable starting a transaction for procedures. The keyword INTERNAL or EXTERNAL must be used to enable or disable this feature.

```
MODULE TXN_CONTROL
LANGUAGE BASIC
PARAMETER COLONS
COMPOUND TRANSACTIONS INTERNAL

PROCEDURE S_TXN (SQLCODE);
BEGIN
SET TRANSACTION READ WRITE;
END;

PROCEDURE C_TXN (SQLCODE);
BEGIN
COMMIT;
END;
```

5.1.15 Undocumented Size Limit for Indexes with Keys Using Collating Sequences

Bug 586079

When a column is defined with a collating sequence, the index key is specially encoded to incorporate the correct ordering (collating) information. This special encoding takes more space than keys encoded for ASCII (the default when no collating sequence is used). Therefore, the encoded string uses more than the customary one byte per character of space within the index. This is true for all versions of Oracle Rdb that support collating sequences.

For all collating sequences, except Norwegian, the space required is approximately 9 bytes for every 8 characters. So, a CHAR (24) column will require approximately 27 bytes. For Norwegian collating sequences, the space required is approximately 10 bytes for every 8 characters.

The space required for encoding the string must be taken into account when calculating the size of an index key against the limit of 255 bytes. Suppose a column defined with a collating sequence of GERMAN was used in an index. The length of that column is limited to a maximum of 225 characters because the key will be encoded in 254 bytes.

The following example demonstrates how a 233 character column, defined with a German collating sequence and included in an index, exceeds the index size limit of 255 bytes, even though the column is defined as less than 255 characters in length:

```
SQL> CREATE DATABASE
cont>     FILENAME 'TESTDB.RDB'
cont>     COLLATING SEQUENCE GERMAN GERMAN;
SQL> CREATE TABLE EMPLOYEE_INFO (
cont>     EMP_NAME CHAR (233));
SQL> CREATE INDEX EMP_NAME_IDX
cont>     ON EMPLOYEE_INFO (
cont>     EMP_NAME     ASC)
cont>     TYPE IS SORTED;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-INDTOOBIG, requested index is too big
```

5.1.16 Changes to RMU/REPLICATE AFTER/BUFFERS Command

The behavior of the RMU/REPLICATE AFTER/BUFFERS command has been changed. The /BUFFERS qualifier may be used with either the CONFIGURE option or the START option.

When using local buffers, the AIJ log roll-forward server (LRS) will use a minimum of 4096 buffers. The value provided to the /BUFFERS qualifier will be accepted, but it will be ignored if it is less than 4096. In addition, further parameters will be checked and the number of buffers may be increased if the resulting calculations are greater than the number of buffers specified by the /BUFFERS qualifier. If the database is configured to use more than 4096 AIJ request blocks (ARBs), then the number of buffers may be increased to the number of ARBs configured for the database. The LRS ensures that there are at least 10 buffers for every possible storage area in the database. Thus, if the total number of storage areas (both used and reserved) multiplied by 10 results in a greater number of buffers, that number will be used.

When global buffers are used, the number of buffers used by the AIJ log roll-forward server is determined as follows:

- If the /BUFFERS qualifier is omitted and the /ONLINE qualifier is specified, the number of buffers will default to the previously configured value, if any, or 256, whichever is larger.
- If the /BUFFERS qualifier is omitted and the /ONLINE qualifier is not specified or the /NOONLINE is specified, the number of buffers will default to the maximum number of global buffers allowed per user ("USER LIMIT"), or 256, whichever is larger.

- If the /BUFFERS qualifier is specified, that value must be at least 256, and it may not be greater than the maximum number of global buffers allowed per user ("USER LIMIT").

The /BUFFER qualifier now enforces a minimum of 256 buffers for the AIJ log roll-forward server. The maximum number of buffers allowed is still 524288 buffers.

5.1.17 Change in the Way RDMAIJ Server is Set Up in UCX

Starting with Oracle Rdb V7.0.2.1, the RDMAIJ image has become a varied image. Therefore, the information in section 2.12, "Step 10: Specify the Network Transport Protocol," of the *Oracle Rdb7 and Oracle CODASYL DBMS Guide to Hot Standby Databases* has become outdated in regards to setting up the RDMAIJSERVER object when using UCX as the network transport protocol. The UCX SET SERVICE command should now look similar to the following:

```
$ UCX SET SERVICE RDMAIJ -
  /PORT=<port_number> -
  /USER_NAME=RDMAIJ -
  /PROCESS_NAME=RDMAIJ -
  /FILE=SYS$SYSTEM:RDMAIJSERVER.com -
  /LIMIT=<limit>
```

And for Oracle Rdb multiversion, it should look similar to the following:

```
$ UCX SET SERVICE RDMAIJ70 -
  /PORT=<port_number> -
  /USER_NAME=RDMAIJ70 -
  /PROCESS_NAME=RDMAIJ70 -
  /FILE=SYS$SYSTEM:RDMAIJSERVER70.com -
  /LIMIT=<limit>
```

The installation procedure for Oracle Rdb creates a user named RDMAIJ(nn) and places a file called RDMAIJSERVER(nn).com in SYS\$SYSTEM and the RMONSTART(nn).COM command procedure will try to enable a service called RDMAIJ(nn) if UCX is installed and running.

Changing the RDMAIJ server to a multivariant image does not impact installations using DECNet since the correct DECNet object is created during the Rdb installation.

5.1.18 CREATE INDEX Supported for Hot Standby

On page 1-13 of the *Guide to Hot Standby Databases*, the add new index operation is incorrectly listed as an offline operation not supported by Hot Standby. The CREATE INDEX operation is now fully supported by Hot Standby, as long as the transaction does not span all available AIJ journals, including emergency AIJ journals.

5.1.19 Dynamic OR Optimization Formats

Bug 711643

In Table C-2 on Page C-7 of the *Oracle Rdb7 Guide to Database Performance and Tuning*, the dynamic OR optimization format is incorrectly documented as [l:h...]n. The correct formats for Oracle Rdb Release 7.0 and later are [(l:h)n] and [(l:h,l2:h2)].

Known Problems and Restrictions

This chapter describes problems, restrictions, and workarounds known to exist in Oracle Rdb7 Release 7.0.6.

6.0.1 Behavior Change in 'With System Logical_Name Translation' Clause

The way logical name translation is performed when 'with system logical_name translation' is specified in the 'location' clause of the 'create function' or the 'create routine' statements has changed. This change occurred between VAX/VMS V5.5-2 and OpenVMS V7.1.

When 'with system logical_name translation' is specified, any logical name in the location string is expanded using only EXECUTIVE_MODE logical names. In VAX/VMS V5.5-2, the logical names are expanded from the SYSTEM logical name table only. In OpenVMS V7.1, the logical names are expanded from the first definition found when searching the logical name tables in (LNMS\$FILE_DEV) order.

Thus, if a logical is only defined in the EXECUTIVE_MODE SYSTEM table (and in no other EXECUTIVE_MODE tables), then there will be no apparent change in behavior. However, if a logical name has been defined in the EXECUTIVE_MODE GROUP table and in the EXECUTIVE_MODE SYSTEM table, then on VAX/VMS V5.5 the SYSTEM table translation will be used and on OpenVMS V7.1 the GROUP table translation will be used.

Oracle believes that this behavioral change is still in keeping with the secure intent of this clause for external routines. An OpenVMS user must have SYSNAM privilege to define an EXEC mode logical in any table. Therefore, it still provides a secure method of locating production sharable images for use by the Rdb server.

A future version of the Oracle Rdb SQL Reference manual will be reworded to remove the reference to the SYSTEM logical name table in the description. The keyword SECURE will be synonymous with SYSTEM in this context.

As an example, if the logical TEST_EXTRTN_1 is defined as:

```
$ show logical/access_mode=executive_mode test_extrtn_1
"TEST_EXTRTN_1" = "NOSUCHIMG9" (LNM$PROCESS_TABLE)
"TEST_EXTRTN_1" = "NOSUCHIMG9" (LNM$JOB_9D277AC0)
"TEST_EXTRTN_1" = "NOSUCHIMGB" (TEST$GROUP_LOGICALS)
"TEST_EXTRTN_1" = "DISK1:[TEST]EXTRTN.EXE" (LNM$SYSTEM_TABLE)
```

Then under VAX/VMS V5.5-2, TEST_EXTRTN_1 will be translated as "DISK1:[TEST]EXTRTN.EXE" whereas under OpenVMS V7.1 it will be translated as "NOSUCHIMG9".

6.0.2 Carry-Over Locks and NOWAIT Transactions Clarification

In NOWAIT transactions, the BLAST mechanism cannot be used. For the blocking user to receive the BLAST signal, the requesting user must request the locked resource with WAIT (which a NOWAIT transaction does not do). Oracle Rdb defines a resource called NOWAIT, which is used to indicate that a NOWAIT transaction has been started. When a NOWAIT transaction starts, the user requests the NOWAIT resource. All other database users hold a lock on the NOWAIT resource so that when the NOWAIT transaction starts, all other users are notified with a NOWAIT BLAST. The BLAST causes blocking users to release any carry-over locks. There can be a delay before the transactions with carry-over locks detect the presence of the NOWAIT transaction and release their carry-over locks. You can detect this condition by examining the stall messages. If the "Waiting for NOWAIT signal (CW)" stall message appears frequently, then the application is probably experiencing a decrease in performance and you should consider disabling the carry-over lock behavior.

6.0.3 Strict Partitioning May Scan Extra Partitions

When you use a WHERE clause with the less than (<) or greater than (>) operator and a value that is the same as the boundary value of a storage map, Oracle Rdb7 scans extra partitions. A boundary value is a value specified in the WITH LIMIT OF clause. The following example, executed while the logical name RDMSS\$DEBUG_FLAGS is defined as "S", illustrates the behavior:

```
ATTACH 'FILENAME MF_PERSONNEL';
CREATE TABLE T1 (ID INTEGER, LAST_NAME CHAR(12), FIRST_NAME CHAR(12));
CREATE STORAGE MAP M FOR T1 PARTITIONING NOT UPDATABLE
STORE USING (ID)
  IN EMPIDS_LOW WITH LIMIT OF (200)
  IN EMPIDS_MID WITH LIMIT OF (400)
  OTHERWISE IN EMPIDS_OVER;
INSERT INTO T1 VALUES (150, 'Boney', 'MaryJean');
INSERT INTO T1 VALUES (350, 'Morley', 'Steven');
INSERT INTO T1 VALUES (300, 'Martinez', 'Nancy');
INSERT INTO T1 VALUES (450, 'Gentile', 'Russ');

SELECT * FROM T1 WHERE ID > 400;
Conjunct      Get      Retrieval sequentially of relation T1
Strict Partitioning: part      2      3
                   ID  LAST_NAME  FIRST_NAME
                   450  Gentile   Russ
1 row selected
```

In the previous example, partition 2 does not need to be scanned. This does not affect the correctness of the result. Users can avoid the extra scan by using values other than the boundary values.

6.0.4 Exclusive Access Transactions May Deadlock With RCS Process

If a table is frequently accessed by long running transactions that request READ /WRITE access reserving the table for EXCLUSIVE WRITE, and if the table has one or more indexes, you may experience deadlocks between the user process and the Row Cache Server (RCS) process.

There are at least three suggested workarounds to this problem:

1. Reserve the table for SHARED WRITE.
2. Close the database and disable row cache for the duration of the exclusive transaction.

3. Change the checkpoint interval for the RCS process to a time longer than the time required to complete the batch job and then trigger a checkpoint just before the batch job starts. Set the interval back to a smaller interval after the checkpoint completes.

6.0.5 Oracle Rdb and OpenVMS ODS-5 Volumes

The OpenVMS Version 7.2 release introduced Extended File Specifications, which consists of two major components:

- A new, optional, volume structure, ODS-5, which provides support for file names that are longer and have a greater range of legal characters than in previous versions of OpenVMS
- Support for deep directories

ODS-5 was introduced primarily to provide enhanced file sharing capabilities for users of Advanced Server for OpenVMS 7.2 (formerly known as PATHWORKS for OpenVMS), as well as DCOM and JAVA applications.

In some cases, Oracle Rdb performs its own file and directory name parsing and explicitly requires ODS-2 (the traditional OpenVMS volume structure) file and directory name conventions to be followed. Because of this knowledge, Oracle does not support any Oracle Rdb database file components (including root files, storage area files, after image journal files, record cache backing store files, database backup files, after image journal backup files, etc.) that utilize any non-ODS-2 file naming features. For this reason, Oracle recommends that Oracle Rdb database components not be located on ODS-5 volumes.

A future release of Oracle Rdb is expected to relax some of these restrictions and support ODS-5 volumes.

6.0.6 Clarification of the USER Impersonation Provided by the Oracle Rdb Server

Bug 551240

In Oracle Rdb V6.1, a new feature was introduced which allowed a user to attach (or connect) to a database by providing a username (USER keyword) and a password (USING keyword). This functionality allows the Rdb Server to impersonate those users in two environments.

- Remote Database Access. When DECnet is used as the remote transport, the Rdb/Dispatch layer of Oracle Rdb uses the provided username and password, or proxy access to create a remote process which matches the named user. However, in a remote connection over TCP/IP, the RDBSERVER process is always logged into RDBSREMOTE rather than a specified user account. In this case the Rdb Server impersonates the user by using the user's UIC (user identification code) during privilege checking. The UIC is assigned by the OpenVMS AUTHORIZE utility.
- SQL/Services database class services. When SQL/Services (possibly accessed by ODBC) accesses a database, it allows the user to logon to the database and the SQL/Services server then impersonates that user in the database.

When a database has access control established using OpenVMS rights identifiers, then access checking in these two environments does not work as expected. For example, if a user JONES was granted the rights identifier PAYROLL_ACCESS, then you would expect a table in the database with SELECT access granted to PAYROLL_ACCESS to be accessible to JONES. This does not

currently work because the Rdb Server does not have the full OpenVMS security profile loaded, just the UIC. So only access granted to JONES is allowed.

This problem results in an error being reported such as the following from ODBC:

```
[Oracle][ODBC][Rdb]%RDB-E-NO_PRIV privileged by database facility (#-1028)
```

This is currently a restriction in this release of Oracle Rdb. In the next major release, support will be provided to inherit the users full security profile into the database.

6.0.7 Index STORE Clause WITH LIMIT OF Not Enforced in Single Partition Map

Bug 413410

An index which has a STORE clause with a single WITH LIMIT OF clause and no OTHERWISE clause doesn't validate the inserted values against the high limit. Normally values beyond the last WITH LIMIT OF clause are rejected during INSERT and UPDATE statements.

Consider this example:

```
create table PTABLE (
  NR
    INTEGER,
  A
    CHAR (2));
create index NR_IDX
  on PTABLE (
  NR)
  type is HASHED
  store using (NR)
  in EMPIDS_LOW
  with limit of (10);
```

When a value is inserted for NR that exceeds the value 10, then an error such as "%RDMS-E-EXCMAPLIMIT, exceeded limit on last partition in storage map for NR_IDX" should be generated. However, this error is only reported if the index has two or more partitions.

A workaround for this problem is to create a CHECK constraint on the column to restrict the upper limit. e.g. CHECK (NR <= 10). This check constraint should be defined as NOT DEFERRABLE and will be solved using an index lookup.

This problem will be corrected in a future version of Oracle Rdb.

6.0.8 Unexpected NO_META_UPDATE Error Generated by DROP MODULE ... CASCADE When Attached by PATHNAME

Bug 755182

The SQL statement DROP MODULE ... CASCADE may sometimes generate an unexpected NO_META_UPDATE error. This occurs when the session attaches to a database by PATHNAME.

```
SQL> drop module m1 cascade;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-OBJ_INUSE, object "M1P1" is referenced by M2.M2P1 (usage: Procedure)
-RDMS-E-MODNOTDEL, module "M1" has not been deleted
```

This error occurs because the CASCADE option is ignored because the Oracle CDD/Repository does not support CASCADE. The workaround is to attach by FILENAME and perform the metadata operation.

In a future version of Oracle Rdb, an informational message will be issued describing the downgrade from CASCADE to RESTRICT in such cases.

6.0.9 Unexpected DATEEQ LILL Error During IMPORT With CREATE INDEX or CREATE STORAGE MAP

Bug 1094071

When the SQL IMPORT statement includes CREATE STORAGE MAP or CREATE INDEX statements which use TIMESTAMP or DATE ANSI literals in the WITH LIMIT OF clause, it fails with the following error:

```
%SQL-F-UNSDATXPR, Unsupported date expression
-SQL-F-DATEEQ LILL, Operands of date/time comparison are incorrect
```

The same CREATE STORAGE MAP or CREATE INDEX statements work correctly when used outside of the IMPORT statement.

This error is generated because the SQL IMPORT statement tries to validate the data type of the column against that of the literal value. However, during this phase of the IMPORT, the table does not yet exist.

A workaround for this problem is to use DATE VMS literals in the WITH LIMIT OF clause and allow the Rdb Server to perform the data type conversion at runtime.

This restriction will be relaxed in a future version of Oracle Rdb.

6.0.10 Application and Oracle Rdb Both Using SYS\$HIBER

In application processes that use Oracle Rdb and the \$HIBER system service (possibly via RTL routines such as LIB\$WAIT), it is important that the application ensures that the event being waited for has actually occurred. Oracle Rdb uses \$HIBER/\$WAKE sequences for interprocess communications particularly when the ALS (AIJ Log Server) or the Row Cache features are enabled.

Oracle Rdb's use of the \$WAKE system service can interfere with other users of \$HIBER (such as the routine LIB\$WAIT) that do not check for event completion, possibly causing a \$HIBER to be unexpectedly resumed without waiting at all.

To avoid these situations, consider altering the application to use a code sequence that avoids continuing without a check for the operation (such as a delay or a timer firing) being complete.

The following pseudo-code shows one example of how a flag can be used to indicate that a timed-wait has completed correctly. The wait does not complete until the timer has actually fired and set TIMER_FLAG to TRUE. This code relies on ASTs being enabled.

```
ROUTINE TIMER_WAIT:
  BEGIN
    ! Clear the timer flag
    TIMER_FLAG = FALSE

    ! Schedule an AST for sometime in the future
    STAT = SYS$SETIMR (TIMADR = DELTATIME, ASTRTN = TIMER_AST)
    IF STAT <> SS$_NORMAL THEN LIB$SIGNAL (STAT)
```

```

! Hibernate. When the $HIBER completes, check to make
! sure that TIMER_FLAG is set indicating that the wait
! has finished.
WHILE TIMER_FLAG = FALSE
DO SYS$HIBER()
END

ROUTINE TIMER_AST:
BEGIN
! Set the flag indicating that the timer has expired
TIMER_FLAG = TRUE

! Wake the main-line code
STAT = SYS$WAKE ()
IF STAT <> SS$_NORMAL THEN LIB$SIGNAL (STAT)
END

```

Starting with OpenVMS V7.1, the LIB\$WAIT routine has been enhanced via the FLAGS argument (with the LIB\$K_NOWAKE flag set) to allow an alternate wait scheme (using the \$\$SYNCH system service) that can avoid potential problems with multiple code sequences using the \$HIBER system service. See the OpenVMS RTL Library (LIB\$) Manual for more information about the LIB\$WAIT routine.

6.0.11 IMPORT Unable to Import Some View Definitions

Bug 520651

View definitions that reference SQL functions, that is functions defined by the CREATE MODULE statement, cannot currently be imported by the SQL IMPORT statement. This is because the views are defined before the functions themselves exist.

The following example shows the errors from IMPORT.

```

IMPORTing view TVIEW
%SQL-F-NOVIERES, unable to import view TVIEW
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-OBSOLETE_METADA, request references metadata objects that no
longer exist
-RDMS-E-RTNNEXTS, routine FORMAT_OUT does not exist in this database
%RDB-E-OBSOLETE_METADA, request references metadata objects that no
longer exist
-RDMS-F-TABNOTDEF, relation TVIEW is not defined in database

```

The following script can be used to demonstrate the problem.

```

create database filename badimp;
create table t (sex char);

create module TFORMAT
language SQL

function FORMAT_OUT (:s char)
returns char(4);
return (case :s
when 'F' then 'Female'
when 'M' then 'Male'
else NULL
end);

end module;

create view TVIEW (m_f) as
select FORMAT_OUT (sex) from t;

commit;

```

```
export database filename badimp into exp;
drop database filename badimp;
import database from exp filename badimp;
```

This restriction will be lifted in a future release of Oracle Rdb. Currently the workaround is to save the view definitions and reapply them after the IMPORT completes.

This restriction does not apply to external functions, created using the CREATE FUNCTION statement, as these database objects are defined before tables and views.

6.0.12 AIJSERVER Privileges

For security reasons, the AIJSERVER account ("RDMAIJSERVER") is created with only NETMBX and TMPMBX privileges. These privileges are sufficient to start Hot Standby, in most cases.

However, for production Hot Standby systems, these privileges are not adequate to ensure continued replication in all environments and workload situations. Therefore, Oracle recommends that the DBA provide the following additional privileges for the AIJSERVER account:

- **ALTPRI**
This privilege allows the AIJSERVER to adjust its own priority to ensure adequate quorum (CPU utilization) to prompt message processing.
- **PSWAPM**
This privilege allows the AIJSERVER to enable and disable process swapping, also necessary to ensure prompt message processing.
- **SETPRV**
This privilege allows the AIJSERVER to temporarily set any additional privileges it may need to access the standby database or its server processes.
- **SYSPRV**
This privilege allows the AIJSERVER to access the standby database rootfile, if necessary.
- **WORLD**
This privilege allows the AIJSERVER to more accurately detect standby database server process failure and handle network failure more reliably.

6.0.13 Lock Remastering and Hot Standby

When using the Hot Standby feature, Oracle recommends that the VMS distributed lock manager resource tree be mastered on the standby node where Hot Standby is started. This can be using any of the following methods:

- **Disable dynamic lock remastering.** This can be done dynamically by setting the SYSGEN parameter PE1 to the value 1.
When using this option, be sure that Hot Standby is started on the node where the standby database is first opened.
- **Increasing the LOCKDIRWT value for the LRS node higher than any other node in the same cluster.** However, this is not a dynamic SYSGEN parameter, and a node re-boot is required.

Failure to prevent dynamic lock remastering may cause severe performance degradation for the standby database, which ultimately may be reflected by decreased master database transaction throughput.

6.0.14 RDB_SETUP Privilege Error

Rdb Web Agent V3.0 exposes a privilege problem with Rdb V7.0 and later. This will be fixed in the next Rdb7 release.

The RDB_SETUP function fails with %RDB-E-NO_PRIV, privilege denied by database facility.

It appears that the only workaround is to give users DBADM privilege. Oracle Corporation does not recommend giving users the DBADM privilege.

6.0.15 Starting Hot Standby on Restored Standby Database May Corrupt Database

If a standby database is modified outside of Hot Standby, then backed up and restored, Hot Standby will appear to start up successfully but will corrupt the standby database. A subsequent query of the database will return unpredictable results, possibly in a bugcheck in DIOFETCH\$FETCH_ONE_LINE. When the standby database is restored from a backup of itself, the database is marked as unmodified. Therefore, Hot Standby cannot tell whether the database had been modified before the backup was taken.

WORKAROUND: None.

6.0.16 Restriction on Compound Statement Nesting Levels

The use of multiple nesting levels of compound statements such as CASE or IF-THEN-ELSE within multistatement procedures can result in excessive memory usage during the compile of the procedure. Virtual memory problems have been reported with 10 or 11 levels of nesting. The following example shows an outline of the type of nesting that can lead to this problem.

```
CREATE MODULE MY_MOD LANGUAGE SQL
PROCEDURE MY PROCEDURE
  ( PARAMETERS .....);

BEGIN
  DECLARE ....;
  SET :VARS = 0;
```

```

SELECT .....;
GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
CASE :FLAG                               ! Case #1
  WHEN 100 THEN SET ...;
  WHEN -811 THEN SET ...;
  WHEN 0 THEN
    SET ...; SELECT ...;
    GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
    CASE :FLAG                             ! Case #2
      WHEN 0 THEN SET ...;
      WHEN -811 THEN SET ...;
      WHEN 100 THEN
        UPDATE...; SET ...;
        GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
        IF :FLAG= 100 THEN SET ...;          ! #1
        ELSE
          IF :FLAG < 0 THEN SET...;          ! #2
        ELSE
          DELETE ...
          GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
          IF :FLAG= 100 THEN SET...;          ! #3
          SET ...;
        ELSE
          IF :FLAG < 0 THEN SET...;          ! #4
        ELSE
          IF IN_CHAR_PARAM = 'S' THEN        ! #5
            UPDATE ...
            GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
            IF :FLAG= 100 THEN SET ...;      ! #6
            ELSE
              IF :FLAG < 0 THEN SET...;      ! #7
              END IF;                          ! #7
            END IF;                             ! #6
          END IF;                               ! #5
        IF :FLAG = 0 THEN                       ! #5
          UPDATE ...
          GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
          IF :FLAG= 100 THEN SET ...;        ! #6
        ELSE
          IF :FLAG < 0 THEN SET ...;          ! #7
        ELSE
          DELETE ...
          GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
          IF :FLAG= 100 THEN SET ...;          ! #8
        ELSE
          IF :FLAG < 0 THEN SET ...;          ! #9
        ELSE
          DELETE ...;
          GET DIAGNOSTICS EXCEPTION 1 :FLAG = RETURNED_SQLCODE;
          IF :FLAG= 100 THEN SET ...;          ! #10
          SET ...;
        ELSE
          IF :FLAG < 0 THEN SET ...;          ! #11
          END IF; (11 end if's for #11 - #1)
        ELSE SET ...;
      END CASE;                                ! Case #2
    ELSE SET ...;
  END CASE;                                    ! Case #1
END;
END MODULE;

```

Workaround: Reduce the complexity of the multistatement procedure. Use fewer levels of compound statement nesting by breaking the multistatement procedure into smaller procedures or by using the CALL statement to execute nested stored procedures.

6.0.17 Back Up All AIJ Journals Before Performing a Hot Standby Switchover Operation

Prior to performing a proper Hot Standby switchover operation from the old master database to the new master database (old standby database), be sure to back up ALL AIJ journals.

If you do not back up the AIJ journals on the old master database prior to switchover, they will be initialized by the Hot Standby startup operation, and you will not have a backup of those AIJ journals.

Failure to back up these journals may place your new master database at risk of not being able to be recovered, requiring another fail-over in the event of system failure.

6.0.18 Concurrent DDL and Read-Only Transaction on the Same Table Not Compatible

It is possible that a read-only transaction could generate a bugcheck at DIOBND\$FETCH_AIP_ENT + 1C4 if there is an active, uncommitted transaction that is making metadata changes to the same table. Analysis shows that the snapshot transaction is picking up stale metadata information. Depending on what metadata modifications are taking place, it is possible for metadata information to be removed from the system tables but still exist in the snapshot file. When the read-only transaction tries to use that information, it no longer exists and causes a bugcheck.

The following example shows the actions of the two transactions:

```
A:          B:
attach
set transaction read write

drop index emp_last_name

          attach
          set transaction read only

          select * from employees
          ...bugcheck...
```

The only workaround is to avoid running the two transactions together.

6.0.19 Oracle Rdb and the SRM_CHECK Tool

The Alpha Architecture Reference Manual, Third Edition (AARM) describes strict rules for using interlocked memory instructions. The Compaq Alpha 21264 (EV6) processor and all future Alpha processors are more stringent than their predecessors in their requirement that these rules be followed. As a result, code that has worked in the past despite noncompliance may now fail when executed on systems featuring the new 21264 processor.

Oracle Rdb Release 7.0.3 supports the Compaq Alpha 21264 (EV6) processor. Oracle has performed extensive testing and analysis of the Rdb code to ensure that it is compliant with the rules for using interlocked memory instructions.

However, customers using the Compaq supplied SRM_CHECK tool may find that several of the Oracle Rdb images cause the tool to report potential alpha architecture violations. Although SRM_CHECK can normally identify a code section in an image by the section's attributes, it is possible for OpenVMS images to contain data sections with those same attributes. As a result, SRM_CHECK may scan data as if it were code, and occasionally, a block of data may look like a noncompliant code sequence. This is the case with the Oracle Rdb supplied images. There is no actual instruction stream violation.

However, customers must use the SRM_CHECK tool on their own application executable image files. It is possible that applications linked with very old version of Oracle Rdb (versions prior to Oracle Rdb Release 6.0-05) could have included illegal interlocked memory instruction sequences produced by very old versions of compilers. This code was included in the Oracle Rdb object library files for some very old versions of Oracle Rdb.

If errant instruction sequences are detected in the objects supplied by the Oracle Rdb object libraries, the correct action is to relink the application with a more-current version of Oracle Rdb.

Additional information about the Compaq Alpha 21264 (EV6) processor interlocked memory instructions issues is available at:

http://www.openvms.digital.com/openvms/21264_considerations.html

6.0.20 Oracle RMU Checksum_Verification Qualifier

The Oracle Rdb RMU BACKUP database backup command includes a Checksum_Verification qualifier.

Specifying Checksum_Verification requests that the RMU Backup command verify the checksum stored on each database page before it is backed up, thereby providing end-to-end error detection on the database I/O.

The Checksum_Verification qualifier uses additional CPU resources but can provide an extra measure of confidence in the quality of the data backed up. Use of the Checksum_Verification qualifier offers an additional level of data security 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.

Note, however, that if you specify the Nochecksum_Verification qualifier, and undetected corruptions exist in your database, the corruptions are included in your backup file and restored when you restore the backup file. Such a corruption might be difficult to recover from, especially if it is not detected until weeks or months after the restore operation is performed.

Oracle Corporation recommends that you use the Checksum_Verification qualifier with all database backup operations because of the improved data integrity this qualifier provides.

Unfortunately, due to an oversight, for versions of Oracle Rdb prior to Version 8.0, the default for online backups is the Nochecksum_Verification qualifier. When you do not specify the Checksum_Verification qualifie on all of your RMU database backup commands.

6.0.21 Do Not Use HYPERSORT with RMU/OPTIMIZE/AFTER_JOURNAL (Alpha)

OpenVMS Alpha V7.1 introduced the high-performance Sort/Merge utility (also known as HYPERSORT). This utility takes advantage of the Alpha architecture to provide better performance for most sort and merge operations.

The high-performance Sort/Merge utility supports a subset of the SOR routines. Unfortunately, the high-performance Sort/Merge utility does not support several of the interfaces used by the RMU/OPTIMIZE/AFTER_JOURNAL command. In addition, the high-performance Sort/Merge utility reports no error or warning when being called with the unsupported options used by the RMU/OPTIMIZE/AFTER_JOURNAL command.

For this reason, the use of the high-performance Sort/Merge utility is not supported for the RMU/OPTIMIZE/AFTER_JOURNAL command. Do not define the logical name SORTSHR to reference HYPERSORT.EXE.

6.0.22 Restriction on Using /NOONLINE with Hot Standby

When a user process is performing a read-only transaction on a standby database, an attempt to start replication on the standby database with the /NOONLINE qualifier will fail with the following error, and the database will be closed clusterwide:

```
%RDMS-F-OPERCLOSE, database operator requested database shutdown
```

In a previous release, the following error was returned and the process doing the read-only transaction was not affected:

```
%RDMS-F-STBYDBINUSE, standby database cannot be exclusively accessed for replication
```

As a workaround, if exclusive access is necessary to the standby database, terminate any user processes before starting replication with the /NOONLINE qualifier.

This restriction is due to another bug fix and will be lifted in a future release of Oracle Rdb.

6.0.23 SELECT Query May Bugcheck with PSII2SCANGETNEXTBBCDUPLICATE Error

Bug 683916

A bugcheck could occur when a ranked B-tree index is used in a query after a database has been upgraded to Release 7.0.1.3. This is a result of index corruption that was introduced in previous versions of Oracle Rdb7. This corruption has been fixed and indexes created using Release 7.0.1.3 will not be impacted.

As a workaround, delete the affected index and re-create it under Oracle Rdb7 Release 7.0.1.3 or later.

6.0.24 DBAPack for Windows 3.1 is Deprecated

Oracle Enterprise Manager DBAPack will no longer be supported for use on Windows 3.1.

6.0.25 Determining Mode for SQL Non-Stored Procedures

Bug 506464.

Although stored procedures allow parameters to be defined with the modes IN, OUT, and INOUT, there is no similar mechanism provided for SQL module language or SQL precompiled procedures. However, SQL still associates a mode with a parameter using the following rules:

Any parameter which is the target of an assignment is considered an OUT parameter. Assignments consist of the following:

- The parameter is assigned a value with the SET or GET DIAGNOSTICS statement. For example:

```
set :p1 = 0;
get diagnostics :p2 = TRANSACTION_ACTIVE;
```

- The parameter is assigned a value with the INTO clause of an INSERT, UPDATE, or SELECT statement. For example:

```
insert into T (col1, col2)
  values (...)
  returning dbkey into :p1;

update accounts
  set account_balance = account_balance + :amount
  where account_number = :p1
  returning account_balance
  into :current_balance;

select last_name
  into :p1
  from employees
  where employee_id = '00164';
```

- The parameter is passed on a CALL statement as an OUT or INOUT argument. For example:

```
begin
call GET_CURRENT_BALANCE (:p1);
end;
```

Any parameter that is the source for a query is considered an IN parameter. Query references include:

- The parameter appears in the SELECT list, WHERE or HAVING clauses of a SELECT, or DELETE statement. For example:

```
select :p1 || last_name, count(*)
  from T
  where last_name like 'Smith%'
  group by last_name
  having count(*) > :p2;

delete from T
  where posting_date < :p1;
```

- The parameter appears on the right side of the assignment in a SET statement or SET clause of an UPDATE statement. For example:

```
set :p1 = (select avg(salary)
  from T
  where department = :p2);

update T
  set col1 = :p1
  where ...;
```

- The parameter is used to provide a value to a column in an INSERT statement. For example:

```
insert into T (col1, col2)
  values (:p1, :p2);
```

- The parameter is referenced by an expression in a TRACE, CASE, IF/ELSEIF, WHILE statement, or by the DEFAULT clause of a variable declaration. For example:

```
begin
declare :v integer default :p1;
DO_LOOP:
while :p2 > :p1
loop
  if :p1 is null then
    leave DO_LOOP;
  end if;
  set :p2 = :p2 + 1;
  ...;
  trace 'Loop at ', :p2;
end loop;
end;
```

- The parameter is passed on a CALL statement as an INOUT or IN argument. For example:

```
begin
call SET_LINE_SPEED (:p1);
end;
```

SQL only copies values from the client (application parameters) to the procedure running in the database server if it is marked as either an IN or INOUT parameter. SQL only returns values from the server to the client application parameter variables if the parameter is an OUT or INOUT parameter.

If a parameter is considered an OUT only parameter, then it must be assigned a value within the procedure, otherwise the result returned to the application is considered undefined. This could occur if the parameter is used within a conditional statement such as CASE or IF/ELSEIF. In the following example, the value returned by :p2 would be undefined if :p1 were negative or zero:

```
begin
if :p1 > 0 then
  set :p2 = (select count(*)
            from T
            where col1 = :p1);
end if;
end;
```

It is the responsibility of the application programmer to ensure that the parameter is correctly assigned values within the procedure. A workaround is to either explicitly initialize the OUT parameter, or make it an INOUT parameter. For example:

```

begin
if :p1 > 0 then
    set :p2 = (select count(*)
              from T
              where coll = :p1);
elseif :p2 is null then
    begin
    end;
end if;
end;

```

The empty statement will include a reference to the parameter to make it an IN parameter as well as an OUT parameter.

6.0.26 DROP TABLE CASCADE Results in %RDB-E-NO_META_UPDATE Error

An error could result when a DROP TABLE CASCADE statement is issued. This occurs when the following conditions apply:

- A table is created with an index defined on the table.
- A storage map is created with a placement via index.
- The storage map is a vertical record partition storage map with two or more STORE COLUMNS clauses.

The error message given is %RDB-E-NO_META_UPDATE, metadata update failed.

The following example shows a table, index, and storage map definition followed by a DROP TABLE CASCADE statement and the resulting error message:

```

SQL> CREATE TABLE VRP_TABLE ( ID INT, ID2 INT);
SQL> COMMIT;
SQL> CREATE UNIQUE INDEX VRP_IDX ON VRP_TABLE (ID)
SQL> STORE IN EMPIDS_LOW;
SQL> COMMIT;
SQL> CREATE STORAGE MAP VRP_MAP
cont> FOR VRP_TABLE
cont> PLACEMENT VIA INDEX VRP_IDX
cont> ENABLE COMPRESSION
cont> STORE COLUMNS (ID)
cont> IN EMPIDS_LOW
cont> STORE COLUMNS (ID2)
cont> IN EMPIDS_MID;
SQL> COMMIT;
SQL>
SQL> DROP TABLE VRP_TABLE CASCADE;
SQL> -- Index VRP_IDX is also being dropped.
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-WISH_LIST, feature not implemented yet
-RDMS-E-VRPINVALID, invalid operation for storage map "VRP_MAP"

```

The workaround to this problem is to first delete the storage map, and then delete the table using the CASCADE option. The following example shows the workaround. The SHOW statement indicates that the table, index, and storage map were deleted:

```

SQL> DROP STORAGE MAP VRP_MAP;
SQL> DROP TABLE VRP_TABLE CASCADE;
SQL> -- Index VRP_IDX is also being dropped.
SQL> COMMIT;
SQL> SHOW TABLE VRP_TABLE
No tables found
SQL> SHOW INDEX VRP_IDX
No indexes found
SQL> SHOW STORAGE MAP VRP_MAP
No Storage Maps Found

```

This problem will be corrected in a future version of Oracle Rdb.

6.0.27 Bugcheck Dump Files with Exceptions at COSI_CHF_SIGNAL

In certain situations, Oracle Rdb bugcheck dump files will indicate an exception at COSI_CHF_SIGNAL. This location is, however, not the address of the actual exception. The actual exception occurred at the previous call frame on the stack (the one listed as the next "Saved PC" after the exception).

For example, consider the following bugcheck file stack information:

```

$ SEARCH RDSBUGCHK.DMP "EXCEPTION", "SAVED PC", "-F-", "-E-"

**** Exception at 00EFA828 : COSI_CHF_SIGNAL + 00000140
%COSI-F-BUGCHECK, internal consistency failure
Saved PC = 00C386F0 : PSIINDEX2JOINSCR + 00000318
Saved PC = 00C0BE6C : PSII2BALANCE + 0000105C
Saved PC = 00C0F4D4 : PSII2INSERTT + 000005CC
Saved PC = 00C10640 : PSII2INSERTTREE + 000001A0
.
.
.

```

In this example, the exception actually occurred at PSIINDEX2JOINSCR offset 00000318. If you have a bugcheck dump with an exception at COSI_CHF_SIGNAL, it is important to note the next "Saved PC" because it will be needed when working with Oracle Rdb Support Services.

6.0.28 Interruptions Possible when Using Multistatement or Stored Procedures

Long running multistatement or stored procedures can cause other users in the database to be interrupted by holding resources needed by those other users. Some resources obtained by the execution of a multistatement or stored procedure will not be released until the multistatement or stored procedure finishes. This problem can be encountered even if the statement contains COMMIT or ROLLBACK statements.

The following example demonstrates the problem. The first session enters an endless loop; the second session attempts to backup the database, but it is permanently interrupted:

Session 1

```
SQL> ATTACH 'FILE MF_PERSONNEL';
SQL> CREATE FUNCTION LIB$WAIT (IN REAL BY REFERENCE)
cont> RETURNS INT;
cont> EXTERNAL NAME LIB$WAIT
cont> LOCATION 'SYS$SHARE:LIBRTL.EXE'
cont> LANGUAGE GENERAL
cont> GENERAL PARAMETER STYLE
cont> VARIANT;
SQL> COMMIT;
SQL> EXIT;
```

```
$ SQL
SQL> ATTACH 'FILE MF_PERSONNEL';
SQL> BEGIN
cont> DECLARE :LAST_NAME LAST_NAME_DOM;
cont> DECLARE :WAIT_STATUS INTEGER;
cont> LOOP
cont> SELECT LAST_NAME INTO :LAST_NAME
cont> FROM EMPLOYEES WHERE EMPLOYEE_ID = '00164';
cont> ROLLBACK;
cont> SET :WAIT_STATUS = LIB$WAIT (5.0);
cont> SET TRANSACTION READ ONLY;
cont> END LOOP;
cont> END;
```

Session 2

```
$ RMU/BACKUP/LOG/ONLINE MF_PERSONNEL MF_PERSONNEL
```

From a third session we can see that the backup process is waiting for a lock held in the first session:

```
$ RMU/SHOW LOCKS /MODE=BLOCKING MF_PERSONNEL
```

```
=====
SHOW LOCKS/BLOCKING Information
=====
```

```
-----
Resource: nowait signal
```

	ProcessID	Process Name	Lock ID	System ID	Requested	Granted
Waiting:	20204383	RMU BACKUP.....	5600A476	00010001	CW	NL
Blocker:	2020437B	SQL.....	3B00A35C	00010001	PR	PR

There is no workaround for this restriction. When the multistatement or stored procedure finishes execution, the resources needed by other processes will be released.

6.0.29 Row Cache Not Allowed on Standby Database While Hot Standby Replication Is Active

The row cache feature may not be active on a Hot Standby database while replication is taking place. The Hot Standby feature will not start if row cache is active on the standby database.

This restriction exists because rows in the row cache are accessed using logical dbkeys. However, information transferred to the Hot Standby database from the after-image journal facility only contains physical dbkeys. Because there is no way to maintain rows in the cache using the Hot Standby processing, the row cache must be disabled on the standby database when the standby database is open and replication is active. The master database is not affected; the row cache feature and the Hot Standby feature may be used together on a master database.

The row cache feature should be identically configured on the master and standby databases in the event failover occurs, but the row cache feature must not be activated on the standby database until it becomes the master.

A new command qualifier, `ROW_CACHE=DISABLED`, has been added to the `RMU/OPEN` command to disable the row cache feature on the standby database. To open the Hot Standby database prior to starting replication, use the `ROW_CACHE=DISABLED` qualifier on the `RMU/OPEN` command.

6.0.30 Hot Standby Replication Waits when Starting if Read-Only Transactions Running

Hot Standby replication will wait to start if there are read-only (snapshot) transactions running on the standby database. The log roll-forward server (LRS) will wait until the read-only transactions commit, and then replication will continue.

This is an existing restriction of the Hot Standby software. This release note is intended to complement the Hot Standby documentation.

6.0.31 Error when Using the `SY$LIBRARY:SQL_FUNCTIONS70.SQL` Oracle Functions Script

If your programming environment is not set up correctly, you may encounter problems running the `SY$LIBRARY:SQL_FUNCTIONS70.SQL` script used to set up the Oracle7 functions being supplied with Oracle Rdb.

The following example shows the error:

```
%RDB-E-EXTFUN_FAIL, external routine failed to compile or execute successfully
-RDMS-E-INVRTNUSE, routine RDB$ORACLE_SQLFUNC_INTR0 can not be used, image
"SQL$FUNCTIONS" not activated
-RDMS-I-TEXT, Error activating image
DISK:[DIR]SQL$FUNCTIONS.;; File not found
```

To resolve this problem, use the `@SY$LIBRARY:RDB$SETVER` to set up the appropriate logical names. This will be necessary for programs that use the functions as well.

In a standard environment, use the command shown in the following example:

```
$ @SY$LIBRARY:RDB$SETVER S
```

In a multiversion environment, use the command shown in the following example:

```
$ @SY$LIBRARY:RDB$SETVER 70
```

6.0.32 DEC C and Use of the `/STANDARD` Switch

Bug 394451

The `SQL$PRE` compiler examines the system to know which dialect of C to generate. That default can be overwritten by using the `/CC=[DECC/VAXC]` switch. The `/STANDARD` switch should not be used to choose the dialect of C.

Support for DEC C was added to the product with V6.0 and this note is meant to clarify that support, not to indicate a change. It is possible to use `/STANDARD=RELAXED_ANSI89` or `/STANDARD=VAXC` correctly, but this is not recommended.

The following example shows both the right and wrong way to compile an Oracle Rdb SQL program. Assume a symbol `SQL$PRE` has been defined, and `DEC C` is the default `C` compiler on the system:

```
$ SQL$PRE/CC ! This is correct.  
$ SQL$PRE/CC=DECC ! This is correct.  
$ SQL$PRE/CC=VAXC ! This is correct.  
  
$ SQL$PRE/CC/STANDARD=VAXC ! This is incorrect.
```

Notice that the `/STANDARD` switch has other options in addition to `RELAXED_ANSI89` and `VAX C`. Those are also not supported.

6.0.33 Excessive Process Page Faults and Other Performance Considerations During Oracle Rdb Sorts

Excessive hard or soft page faulting can be a limiting factor of process performance. Sometimes this page faulting occurs during Oracle Rdb sort operations. This note describes how page faulting can occur and some ways to help control, or at least understand, it.

One factor contributing to Oracle Rdb process page faulting is sorting operations. Common causes of sorts include the `SQL GROUP BY`, `ORDER BY`, `UNION`, and `DISTINCT` clauses specified for query and index creation operations. Defining the logical name `RDMS$DEBUG_FLAGS` to "RS" can help determine when Oracle Rdb sort operations are occurring and to display the sort keys and statistics.

Oracle Rdb includes its own copy of the OpenVMS `SORT32` code within the Oracle Rdb images and does not generally call the routines in the OpenVMS run-time library. A copy of the `SORT32` code is used to provide stability between versions of Oracle Rdb and OpenVMS and because Oracle Rdb calls the sort routines from executive processor mode which is difficult to do using the `SORT32` sharable image. Database import and `RMU` load operations call the OpenVMS sort run-time library.

At the beginning of a sort operation, the sort code allocates some memory for working space. The sort code uses this space for buffers, in-memory copies of the data, and sorting trees.

Sort code does not directly consider the process quotas or parameters when allocating memory. The effects of `WSQUOTA` and `WSEXTENT` are indirect. At the beginning of each sort operation, the sort code attempts to adjust the process' working set to the maximum possible size using the `$ADJWSL` system service specifying a requested working set limit of `%X7FFFFFFF` pages (the maximum possible). Sort code then uses a value of 75% of the returned working set for virtual memory scratch space. The scratch space is then initialized and the sort begins.

The initialization of the scratch space generally causes page faults to access the pages newly added to the working set. Pages that were in the working set already may be faulted out as new pages are faulted in. Once the sort operation completes, the pages that may have been faulted out of the working set are likely to be faulted back into the working set.

When a process' working set is limited by the working set quota (`WSQUOTA`) parameter and the working set extent (`WSEXTENT`) parameter is a much larger value, the first call to the sort routines can cause many page faults as the working set grows. Using a value of `WSEXTENT` that is closer to `WSQUOTA` can help reduce the impact of this case.

With some OpenVMS versions, AUTOGEN sets the SYSGEN parameter PQL_MWSEXTENT equal to the WSMAX parameter. This means that all processes on the system end up with WSEXTENT the same as WSMAX. Because WSMAX might be quite high, sorting might result in excessive page faulting. You may want to explicitly set PQL_MWSEXTENT to a lower value if this is the case on your system.

Sort work files are another factor to consider when tuning Oracle Rdb sort operations. When the operation cannot be done in available memory, sort code will use temporary disk files to hold the data as it is being sorted. The *Oracle Rdb Guide to Performance and Tuning* contains more detailed information about sort work files.

The logical name RDMS\$BIND_SORT_WORKFILES specifies how many work files sort code is to use if work files are required. The default is 2, and the maximum number is 10. The work files can be individually controlled by the SORTWORKn logical names (where n is from 0 through 9). You can increase the efficiency of sort operations by assigning the location of the temporary sort work files to different disks. These assignments are made by using up to 10 logical names, SORTWORK0 through SORTWORK9.

Normally, sort code places work files in the user's SYSSCRATCH directory. By default, SYSSCRATCH is the same device and directory as the SYS\$LOGIN location. Spreading the I/O load over many disks improves efficiency as well as performance by taking advantage of the system resources and helps prevent disk I/O bottlenecks. Specifying that a user's work files will reside on separate disks permits overlap of the sort read/write cycle. You may also encounter cases where insufficient space exists on the SYSSCRATCH disk device, such as when Oracle Rdb builds indexes for a very large table. Using the SORTWORK0 through SORTWORK9 logical names can help you avoid this problem.

Note that sort code uses the work files for different sorted runs, and then merges the sorted runs into larger groups. If the source data is mostly sorted, then not every sort work file may need to be accessed. This is a possible source of confusion because even with 10 sort work files, it is possible to exceed the capacity of the first sort file, and the sort operation will fail never having accessed the remaining 9 sort work files.

Note that the logical names RDMS\$BIND_WORK_VM and RDMS\$BIND_WORK_FILE do not affect or control the operation of sort. These logical names are used to control other temporary space allocations within Oracle Rdb.

6.0.34 Performance Monitor Column Mislabeled

The File IO Overview statistics screen, in the Rdb Performance Monitor, contains a column labeled Pages Checked. The column should be labeled Pages Discarded to correctly reflect the statistic displayed.

6.0.35 Restriction Using Backup Files Created Later than Oracle Rdb7 Release 7.0.1

Bug 521583

Backup files created using Oracle Rdb7 releases later than 7.0.1 cannot be restored using Oracle Rdb7 Release 7.0.1. To fix a problem in a previous release, some internal backup file data structures were changed. These changes are not backward compatible with Oracle Rdb7 Release 7.0.1.

If you restore the database using such a backup file, then any attempt to access the restored database may result in unpredictable behavior, even though a verify operation may indicate no problems.

There is no workaround to this problem. For this reason, Oracle Corporation strongly recommends performing a full and complete backup both before and after the upgrade from Release 7.0.1 to later releases of Oracle Rdb7.

6.0.36 RMU Backup Operations and Tape Drive Types

When using more than one tape drive for an RMU backup operation, all the tape drives must be of the same type. For example, all the tape drives must be either TA90s or TZ87s or TK50s. Using different tape drive types (one TK50 and one TA90) for a single database backup operation may make database restoration difficult or impossible.

Oracle RMU attempts to prevent using different tape drive densities during a backup operation, but is not able to detect all invalid cases and expects that all tape drives for a backup are of the same type.

As long as all the tapes used during a backup operation can be read by the same type of tape drive during a restore operation, the backup is likely to be valid. This may be the case, for example, when using a TA90 and a TA90E.

Oracle recommends that, on a regular basis, you test your backup and recovery procedures and environment using a test system. You should restore the databases and then recover them using AIJs to simulate failure recovery of the production system.

Consult the *Oracle Rdb Guide to Database Maintenance*, the *Oracle Rdb Guide to Database Design and Definition*, and the *Oracle RMU Reference Manual* for additional information about Oracle Rdb backup and restore operations.

6.0.37 Use of Oracle Rdb from Shared Images

Bug 470946

If code in the image initialization routine of a shared image makes any calls into Oracle Rdb, through SQL or any other means, access violations or other unexpected behavior may occur if Oracle Rdb's images have not had a chance to do their own initialization.

To avoid this problem, applications must do one of the following:

- Do not make Oracle Rdb calls from the initialization routines of shared images.
- Link in such a way that the RDBSHR.EXE image initializes first. This can be done by placing the reference to RDBSHR.EXE and any other Oracle Rdb shared images last in the linker options file.

6.0.38 Interactive SQL Command Line Editor Rejects Eight-Bit Characters

Digital UNIX Systems

The interactive SQL command line editor on Digital UNIX can interfere with entering eight-bit characters from the command line. The command line editor assumes that a character with the eighth bit set will invoke an editing function. If the command line editor is enabled and a character with the eighth bit set is entered from the command line, the character will not be inserted on the command line. If the character has a corresponding editor function, the function will be invoked; otherwise, the character is considered invalid and rejected.

There are two ways to enter eight-bit characters from the SQL command line: either disable the command line editor or use the command line editor character quoting function to enter each eight-bit character. To disable the command line editor, set the configuration parameter RDB_NOLINEDIT in the configuration file. For example:

```
! Disable the interactive SQL command line editor.
RDB_NOLINEDIT ON
```

To place quotation marks around a character using the command line editor, type Ctrl/V before each character to be placed in quotation marks.

6.0.39 Restriction Added for CREATE STORAGE MAP on Table with Data

Oracle Rdb7 added support that allows a storage map to be added to an existing table which contains data. The restrictions listed for Oracle Rdb7 were:

- The storage map must be a simple map that references only the default storage area and represents the current (default) mapping for the table. The default storage area is either RDB\$SYSTEM or the area name provided by the CREATE DATABASE...DEFAULT STORAGE AREA clause.
- The new map cannot change THRESHOLDS or COMPRESSION for the table, nor can it use the PLACEMENT VIA INDEX clause. It can only contain one area and cannot be vertically partitioned. This new map simply describes the mapping as it exists by default for the table.

This release of Rdb7 adds the additional restriction that the storage map may not include a WITH LIMIT clause for the storage area. The following example shows the reported error:

```
SQL> CREATE TABLE MAP_TEST1 (A INTEGER, B CHAR(10));
SQL> CREATE INDEX MAP_TEST1_INDEX ON MAP_TEST1 (A);
SQL> INSERT INTO MAP_TEST1 (A, B) VALUES (3, 'Third');
1 row inserted
SQL> CREATE STORAGE MAP MAP_TEST1_MAP FOR MAP_TEST1
cont> STORE USING (A) IN RDB$SYSTEM
cont> WITH LIMIT OF (10); -- can't use WITH LIMIT clause
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-RELNOTEEMPTY, table "MAP_TEST1" has data in it
-RDMS-E-NOCMPLXMAP, can not use complex map for non-empty table
```

6.0.40 ALTER DOMAIN...DROP DEFAULT Reports DEFVALUNS Error

Bug 456867

If a domain has a DEFAULT of CURRENT_USER, SESSION_USER, or SYSTEM_USER and attempts to delete that default, it may fail unexpectedly. The following example shows the error:

```
SQL> ATTACH 'FILENAME PERSONNEL';
SQL> CREATE DOMAIN ADDRESS_DATA2_DOM CHAR(31)
cont> DEFAULT CURRENT_USER;
SQL> COMMIT;
SQL> ALTER DOMAIN ADDRESS_DATA2_DOM
cont> DROP DEFAULT;
%SQL-F-DEFVALUNS, Default values are not supported for the data type of
ADDRESS_DATA2_DOM
```

To work around this problem you must first alter the domain to have a default of NULL, as shown, and then use DROP DEFAULT:

```
SQL> ALTER DOMAIN ADDRESS_DATA2_DOM
cont> SET DEFAULT NULL;
SQL> ALTER DOMAIN ADDRESS_DATA2_DOM
cont> DROP DEFAULT;
SQL> COMMIT;
```

This problem will be corrected in a future release of Oracle Rdb.

6.0.41 Oracle Rdb7 Workload Collection Can Stop Hot Standby Replication

If you are replicating your Oracle Rdb7 database using the Oracle Hot Standby option, you must not use the workload collection option. By default, workload collection is disabled. However, if you enabled workload collection, you must disable it on the master database prior to performing a backup operation on that master database if it will be used to create the standby database for replication purposes. If you do not disable workload collection, it could write workload information to the standby database and prevent replication operations from occurring.

The workaround included at the end of this section describes how to disable workload collection on the master database and allow the Hot Standby software to propagate the change to the standby database automatically during replication operations.

Background Information

By default, workload collection and cardinality collection are automatically disabled when Hot Standby replication operations are occurring on the standby database. However, if replication stops (even for a brief network failure), Oracle Rdb7 potentially can start a read/write transaction on the standby database to write workload collection information. Then, because the standby database is no longer synchronized transactionally with the master database, replication operations cannot restart.

Note

The Oracle Rdb7 optimizer can update workload collection information in the RDB\$WORKLOAD system table even though the standby database is opened exclusively for read-only queries. A read/write transaction is started during the disconnection from the standby database to flush the workload and cardinality statistics to the system tables.

If the standby database is modified, you receive the following messages when you try to restart Hot Standby replication operations:

```
%RDMS-F-DBMODIFIED, database has been modified; AIJ roll-forward not possible
%RMU-F-FATALRDB, Fatal error while accessing Oracle Rdb.
```

Workaround

To work around this problem, perform the following:

- On the master database, disable workload collection using the SQL clause **WORKLOAD COLLECTION IS DISABLED** on the ALTER DATABASE statement. For example:

```
SQL> ALTER DATABASE FILE mf_personnel
cont> WORKLOAD COLLECTION IS DISABLED;
```

This change is propagated to the standby database automatically when you restore the standby database and restart replication operations. Note that, by default, the workload collection feature is disabled. You need to disable workload collection only if you previously enabled workload collection with the `WORKLOAD COLLECTION IS ENABLED` clause.

- On the standby database, include the `Transaction_Mode` qualifier on the `RMU/Restore` command when you restore the standby database. You should set this qualifier to read-only to prevent modifications to the standby database when replication operations are not active. The following example shows the `Transaction_Mode` qualifier used in a typical `RMU/Restore` command:

```
$ RMU/RESTORE /TRANSACTION_MODE=READ_ONLY
              /NOADD
              /NOLOG
              /ROOT=DISK1:[DIR]standby_personnel.rdb
              /AIJ_OPT=aij_opt.dat
              DISK1:[DIR]standby_personnel.rbf
```

If, in the future, you fail over processing to the standby database (so that the standby database becomes the master database), you can re-enable updates to the “new” master database. For example, to re-enable updates, use the SQL statement `ALTER DATABASE` and include the `SET TRANSACTION MODES (ALL)` clause. The following example shows this statement used on the new master database:

```
SQL> ALTER DATABASE FILE mf_personnel
cont> SET TRANSACTION MODES (ALL);
```

6.0.42 RMU Convert Command and System Tables

When the `RMU Convert` command converts a database from a previous version to Oracle Rdb V7.0 or higher, it sets the `RDB$CREATED` and `RDB$LAST_ALTERED` columns to the timestamp of the convert operation.

The `RDB$xxx_CREATOR` columns are set to the current user name (which is space filled) of the converter. Here `xxx` represents the object name, such as in `RDB$TRIGGER_CREATOR`.

The `RMU Convert` command also creates the new index on `RDB$TRANSFER_RELATIONS` if the database is transfer enabled.

6.0.43 Converting Single-File Databases

Because of a substantial increase in the database root file information for Release 7.0, you should ensure that you have adequate disk space before you use the `RMU Convert` command with single-file databases and Release 7.0 or higher.

The size of the database root file of any given database will increase a minimum of 13 blocks and a maximum of 597 blocks. The actual increase depends mostly on the maximum number of users specified for the database.

6.0.44 Restriction when Adding Storage Areas with Users Attached to Database

If you try to interactively add a new storage area where the page size is smaller than the smallest existing page size and the database has been manually opened or users are active, the add operation fails with the following error:

```
%RDB-F-SYS_REQUEST, error from system services request
-RDMS-F-FILACCERR, error opening database root DKA0:[RDB]TEST.RDB;1
-SYSTEM-W-ACCONFLICT, file access conflict
```

You can make this change only when no users are attached to the database and, if the database is set to OPEN IS MANUAL, the database is closed. Several internal Oracle Rdb data structures are based on the minimum page size, and these structures cannot be resized if users are attached to the database.

Furthermore, because this particular change is not recorded in the AIJ file, any recovery scenario will fail. Note also that if you use .aij files, you must backup the database and restart after-image journaling because this change invalidates the current AIJ recovery.

6.0.45 Restriction on Tape Usage for Digital UNIX V3.2

Digital UNIX Systems

You can experience a problem where you are unable to use multiple tapes with the Oracle RMU Backup command with Digital UNIX V3.2. Every attempt to recover fails. If this happens and device errors are logged in the system error log, it is possible that the operation succeeded, but the device open reference count is zeroed out. This means that any attempt to use the drive by the process holding the open file descriptor will fail with EINVAL status but another process will be able to open and use the drive even while the first process has it opened.

There is no workaround for this problem. This problem with the magtape driver will be corrected in a future release of Digital UNIX.

6.0.46 Support for Single-File Databases to be Dropped in a Future Release

Oracle Rdb currently supports both single-file and multifile databases on both OpenVMS and Digital UNIX. However, single-file databases will not be supported in a future release of Oracle Rdb. At that time, Oracle Rdb will provide the means to easily convert single-file databases to multifile databases.

Oracle recommends that users with single-file databases perform the following actions:

- Use the Oracle RMU commands, such as Backup and Restore, to make copies, back up, or move single-file databases. Do not use operating system commands to copy, back up, or move databases.
- Create new databases as multifile databases even though single-file databases are supported in Oracle Rdb release 6.1 and release 7.0.

6.0.47 DECdtm Log Stalls

Resource managers using the DECdtm services can sometimes suddenly stop being able to commit transactions. If Oracle Rdb7 is installed and transactions are being run, an RMU Show command on the affected database will show transactions as being "stalled, waiting to commit".

Refer to the DECdtm documentation and release notes for information on symptoms, fixes, and workarounds for this problem. One workaround, for OpenVMS V5.5-x, is provided here.

On the affected node while the log stall is in progress, type the following command from a privileged account:

```
$ MCR LMCP SET NOTIMEZONE
```

This should force the log to restart.

This stall occurs only when a particular bit in a pointer field becomes set. To see the value of the pointer field, enter the following command from a privileged account (where <nodename> is the SCS node name of the node in question).

```
$ MCR LMCP DUMP/ACTIVE/NOFORM SYSTEM$<nodename>
```

This command displays output similar to the following:

```
Dump of transaction log SYS$COMMON:[SYSEXE]SYSTEM$<nodename>.LM$JOURNAL;1
End of file block 4002 / Allocated 4002
Log Version 1.0
Transaction log UID: 29551FC0-CBB7-11CC-8001-AA000400B7A5
Penultimate Checkpoint: 000013FD4479 0079
Last Checkpoint: 000013FD4479 0084

Total of 2 transactions active, 0 prepared and 2 committed.
```

The stall will occur when bit 31 of the checkpoint address becomes set, as this excerpt from the previous example shows:

```
Last Checkpoint: 000013FD4479 0084
                   ^
                   |
```

When the number indicated in the example becomes 8, the log will stall. Check this number and observe how quickly it grows. When it is at 7FFF, frequently use the following command:

```
$ MCR LMCP SHOW LOG /CURRENT
```

If this command shows a stall in progress, use the workaround to restart the log.

See your Compaq Computer Corporation representative for information about patches to DECdtm.

6.0.48 Cannot Run Distributed Transactions on Systems with DECnet/OSI and OpenVMS Alpha Version 6.1 or OpenVMS VAX Version 6.0

If you have DECnet/OSI installed on a system with OpenVMS Alpha Version 6.1 or OpenVMS VAX Version 6.0, you cannot run Oracle Rdb7 operations that require the two-phase commit protocol. The two-phase commit protocol guarantees that if one operation in a distributed transaction cannot be completed, none of the operations is completed.

If you have DECnet/OSI installed on a system running OpenVMS VAX Version 6.1 or higher or OpenVMS Alpha Version 6.2 or higher, you can run Oracle Rdb operations that require the two-phase commit protocol.

For more information about the two-phase commit protocol, see the *Oracle Rdb Guide to Distributed Transactions*.

6.0.49 Multiblock Page Writes May Require Restore Operation

If a node fails while a multiblock page is being written to disk, the page in the disk becomes inconsistent and is detected immediately during failover. (Failover is the recovery of an application by restarting it on another computer.) The problem is rare and occurs because only single-block I/O operations are guaranteed by OpenVMS to be written atomically. This problem has never been reported by any customer and was detected only during stress tests in our labs.

Correct the page by an area-level restore operation. Database integrity is not compromised, but the affected area will not be available until the restore operation completes.

A future release of Oracle Rdb will provide a solution that guarantees multiblock atomic write operations. Cluster failovers will automatically cause the recovery of multiblock pages, and no manual intervention will be required.

6.0.50 Oracle Rdb7 Network Link Failure Does Not Allow DISCONNECT to Clean Up Transactions

If a program attaches to a database on a remote node and it loses the connection before the COMMIT statement is issued, there is nothing you can do except exit the program and start again.

The problem occurs when a program is connected to a remote database and updates the database, but then just before it commits, the network fails. When the commit executes, SQL shows, as it normally should, that the program has lost the link. Assume that the user waits for a minute or two, then tries the transaction again. The problem is that when the start transaction is issued for the second time, it fails because old information still exists about the previous failed transaction. This occurs even if the user issues a DISCONNECT statement (in Release 4.1 and earlier, a FINISH statement), which also fails with an RDB-E-IO_ERROR error message.

6.0.51 Replication Option Copy Processes Do Not Process Database Pages Ahead of an Application

When a group of copy processes initiated by the Replication Option (formerly Data Distributor) begins running after an application has begun modifying the database, the copy processes will catch up to the application and will not be able to process database pages that are logically ahead of the application in the RDB\$CHANGES system table. The copy processes all align waiting for the same database page and do not move on until the application has released it. The performance of each copy process degrades because it is being paced by the application.

When a copy process completes updates to its respective remote database, it updates the RDB\$TRANSFERS system table and then tries to delete any RDB\$CHANGES rows not needed by any transfers. During this process, the RDB\$CHANGES table cannot be updated by any application process, holding up any database updates until the deletion process is complete. The application stalls while waiting for the RDB\$CHANGES table. The resulting contention for RDB\$CHANGES SPAM pages and data pages severely impacts performance throughput, requiring user intervention with normal processing.

This is a known restriction in Release 4.0 and higher. Oracle Rdb uses page locks as latches. These latches are held only for the duration of an action on the page and not to the end of transaction. The page locks also have blocking asynchronous system traps (ASTs) associated with them. Therefore, whenever a process requests a page lock, the process holding that page lock is sent a blocking AST (BLAST) by OpenVMS. The process that receives such a blocking AST queues the fact that the page lock should be released as soon as possible. However, the page lock cannot be released immediately.

Such work requests to release page locks are handled at verb commit time. An Oracle Rdb verb is an Oracle Rdb query that executes atomically, within a transaction. Therefore, verbs that require the scan of a large table, for example, can be quite long. An updating application does not release page locks until its verb has completed.

The reasons for holding on to the page locks until the end of the verb are fundamental to the database management system.

6.0.52 SQL Does Not Display Storage Map Definition After Cascading Delete of Storage Area

When you delete a storage area using the CASCADE keyword and that storage area is not the only area to which the storage map refers, the SHOW STORAGE MAP statement no longer shows the placement definition for that storage map.

The following example demonstrates this restriction:

```
SQL> SHOW STORAGE MAP DEGREES_MAP1
DEGREES_MAP1
For Table:           DEGREES1
Compression is:     ENABLED
Partitioning is:    NOT UPDATABLE
Store clause:       STORE USING (EMPLOYEE_ID)
                    IN DEG_AREA WITH LIMIT OF ('00250')
                    OTHERWISE IN DEG_AREA2

SQL> DISCONNECT DEFAULT;
SQL> -- Drop the storage area, using the CASCADE keyword.
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> DROP STORAGE AREA DEG_AREA CASCADE;
SQL> --
SQL> -- Display the storage map definition.
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SHOW STORAGE MAP DEGREES_MAP1
DEGREES_MAP1
For Table:           DEGREES1
Compression is:     ENABLED
Partitioning is:    NOT UPDATABLE

SQL>
```

The other storage area, DEG_AREA2, still exists, even though the SHOW STORAGE MAP statement does not display it.

A workaround is to use the RMU Extract command with the Items=Storage_Map qualifier to see the mapping.

6.0.53 ARITH_EXCEPT or Incorrect Results Using LIKE IGNORE CASE

When you use LIKE . . . IGNORE CASE, programs linked under Oracle Rdb Release 4.2 and Release 5.1, but run under higher versions of Oracle Rdb, may result in incorrect results or %RDB-E-ARITH_EXCEPT exceptions.

To work around the problem, avoid using IGNORE CASE with LIKE, or recompile and relink under a higher version (Release 6.0 or higher.)

6.0.54 Different Methods of Limiting Returned Rows from Queries

You can establish the query governor for rows returned from a query by using the SQL SET QUERY LIMIT statement, a logical name, or a configuration parameter. This note describes the differences between the mechanisms.

- If you define the RDMS\$BIND_QG_REC_LIMIT logical name or RDB_BIND_QG_REC_LIMIT configuration parameter to a small value, the query will often fail with no rows returned. The following example demonstrates setting the limit to 10 rows and the resulting failure:

```

$ DEFINE RDMS$BIND_QG_REC_LIMIT 10
$ SQL$
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES;
%RDB-F-EXQUOTA, Oracle Rdb runtime quota exceeded
-RDMS-E-MAXRECLIM, query governor maximum limit of rows has been reached

```

Interactive SQL must load its metadata cache for the table before it can process the SELECT statement. In this example, interactive SQL loads its metadata cache to allow it to check that the column EMPLOYEE_ID really exists for the table. The queries on the Oracle Rdb system tables RDB\$RELATIONS and RDB\$RELATION_FIELDS exceed the limit of rows.

Oracle Rdb does not prepare the SELECT statement, let alone execute it. Raising the limit to a number less than 100 (the cardinality of EMPLOYEES) but more than the number of columns in EMPLOYEES (that is, the number of rows to read from the RDB\$RELATION_FIELDS system table) is sufficient to read each column definition.

To see an indication of the queries executed against the system tables, define the RDMS\$DEBUG_FLAGS logical name or the RDB_DEBUG_FLAGS configuration parameter as S or B.

- If you set the row limit using the SQL SET QUERY statement and run the same query, it returns the number of rows specified by the SQL SET QUERY statement before failing:

```

SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SET QUERY LIMIT ROWS 10;
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES;
EMPLOYEE_ID
00164
00165
.
.
.
00173
%RDB-E-EXQUOTA, Oracle Rdb runtime quota exceeded
-RDMS-E-MAXRECLIM, query governor maximum limit of rows has been reached

```

The SET QUERY LIMIT specifies that only user queries be limited to 10 rows. Therefore, the queries used to load the metadata cache are not restricted in any way.

Like the SET QUERY LIMIT statement, the SQL precompiler and module processor command line qualifiers (QUERY_MAX_ROWS and SQLOPTIONS=QUERY_MAX_ROWS) only limit user queries.

Keep the differences in mind when limiting returned rows using the logical name RDMS\$BIND_QG_REC_LIMIT or the configuration parameter RDB_BIND_QG_REC_LIMIT. They may limit more queries than are obvious. This is important when using 4GL tools, the SQL precompiler, the SQL module processor, and other interfaces that read the Oracle Rdb system tables as part of query processing.

6.0.55 Suggestions for Optimal Usage of the SHARED DATA DEFINITION Clause for Parallel Index Creation

The CREATE INDEX process involves the following steps:

1. Process the metadata.
2. Lock the index name.

Because new metadata (which includes the index name) is not written to disk until the end of the index process, Oracle Rdb must ensure index name uniqueness across the database during this time by taking a special lock on the provided index name.

3. Read the table for sorting by selected index columns and ordering.
4. Sort the key data.
5. Build the index (includes partitioning across storage areas).
6. Write new metadata to disk.

Step 6 is the point of conflict with other index definers because the system table and indexes are locked like any other updated table.

Multiple users can create indexes on the same table by using the `RESERVING table_name FOR SHARED DATA DEFINITION` clause of the `SET TRANSACTION` statement. For optimal usage of this capability, Oracle Rdb suggests the following guidelines:

- You should commit the transaction immediately after the `CREATE INDEX` statement so that locks on the table are released. This avoids lock conflicts with other index definers and improves overall concurrency.
- By assigning the location of the temporary sort work files `SORTWORK0`, `SORTWORK1`, . . . , `SORTWORK9` to different disks for each parallel process that issues the `SHARED DATA DEFINITION` statement, you can increase the efficiency of sort operations. This minimizes any possible disk I/O bottlenecks and allows overlap of the `SORT` read/write cycle.
- If possible, enable global buffers and specify a buffer number large enough to hold a sufficient amount of table data. However, do not define global buffers larger than the available system physical memory. Global buffers allow sharing of database pages and thus result in disk I/O savings. That is, pages are read from disk by one of the processes and then shared by the other index definers for the same table, reducing the I/O load on the table.
- If global buffers are not used, ensure that enough local buffers exist to keep much of the index cached (use the `RDM$BIND_BUFFERS` logical name or `RDB_BIND_BUFFERS` configuration parameter or the `NUMBER OF BUFFERS IS` clause in `SQL` to change the number of buffers).
- To distribute the disk I/O load, place the storage areas for the indexes on separate disk drives. Note that using the same storage area for multiple indexes will result in contention during the index creation (Step 5) for `SPAM` pages.
- Consider placing the `.ruj` file for each parallel definer on its own disk or an infrequently used disk.
- Even though snapshot I/O should be minimal, consider disabling snapshots during parallel index creation.
- Refer to the *Oracle Rdb Guide to Performance and Tuning* to determine the appropriate working set values for each process to minimize excessive paging activity. In particular, avoid using working set parameters where the difference between `WSQUOTA` and `WSEXTENT` is large. The `SORT` utility uses the difference between these two values to allocate scratch virtual memory. A large difference (that is, the requested virtual memory grossly exceeds the available physical memory) may lead to excessive page faulting.

- The performance benefits of using SHARED DATA DEFINITION can best be observed when creating many indexes in parallel. The benefit is in the average elapsed time, not in CPU or I/O usage. For example, when two indexes are created in parallel using the SHARED DATA DEFINITION clause, the database must be attached twice, and the two attaches each use separate system resources.
- Using the SHARED DATA DEFINITION clause on a single-file database or for indexes defined in the RDB\$SYSTEM storage area is not recommended.

The following table displays the elapsed time benefit when creating multiple indexes in parallel with the SHARED DATA DEFINITION clause. The table shows the elapsed time for 10 parallel process index creations (Index1, Index2, . . . Index10) and one process with 10 sequential index creations (All10). In this example, global buffers are enabled and the number of buffers is 500. The longest time for a parallel index creation is Index7 with an elapsed time of 00:02:34.64, compared to creating 10 indexes sequentially with an elapsed time of 00:03:26.66. The longest single parallel create index elapsed time is shorter than the elapsed time of creating all 10 of the indexes serially.

Index Create Job	Elapsed Time
Index1	00:02:22.50
Index2	00:01:57.94
Index3	00:02:06.27
Index4	00:01:34.53
Index5	00:01:51.96
Index6	00:01:27.57
Index7	00:02:34.64
Index8	00:01:40.56
Index9	00:01:34.43
Index10	00:01:47.44
All 10	00:03:26.66

6.0.56 Side Effect when Calling Stored Routines

When calling a stored routine, you must not use the same routine to calculate argument values by a stored function. For example, if the routine being called is also called by a stored function during the calculation of an argument value, passed arguments to the routine may be incorrect.

The following example shows a stored procedure P being called during the calculation of the arguments for another invocation of the stored procedure P:

```

SQL> CREATE MODULE M
cont>     LANG SQL
cont>
cont>     PROCEDURE P (IN :A INTEGER, IN :B INTEGER, OUT :C INTEGER);
cont>     BEGIN
cont>     SET :C = :A + :B;
cont>     END;
cont>
cont>     FUNCTION F () RETURNS INTEGER
cont>     COMMENT IS 'expect F to always return 2';
cont>     BEGIN
cont>     DECLARE :B INTEGER;
cont>     CALL P (1, 1, :B);
cont>     TRACE 'RETURNING ', :B;
cont>     RETURN :B;
cont>     END;
cont> END MODULE;
SQL>
SQL> SET FLAGS 'TRACE';
SQL> BEGIN
cont> DECLARE :CC INTEGER;
cont> CALL P (2, F(), :CC);
cont> TRACE 'Expected 4, got ', :CC;
cont> END;
~Xt: returning 2
~Xt: Expected 4, got 3

```

The result as shown above is incorrect. The routine argument values are written to the called routine's parameter area before complex expression values are calculated. These calculations may (as in the example) overwrite previously copied data.

The workaround is to assign the argument expression (in this example calling the stored function F) to a temporary variable and pass this variable as the input for the routine. The following example shows the workaround:

```

SQL> BEGIN
cont> DECLARE :BB, :CC INTEGER;
cont> SET :BB = F();
cont> CALL P (2, :BB, :CC);
cont> TRACE 'Expected 4, got ', :CC;
cont> END;
~Xt: returning 2
~Xt: Expected 4, got 4

```

This problem will be corrected in a future version of Oracle Rdb7.

6.0.57 Nested Correlated Subquery Outer References Incorrect

This problem was corrected in Oracle Rdb7 Release 7.0.0.2. An updated release note stating that this was fixed was inadvertently left out of all the following sets of release notes. Please note that this issue is now corrected. Outer references from aggregation subqueries contained within nested queries could receive incorrect values, causing the overall query to return incorrect results. The general symptom for an outer query that returned rows 1 to n was that the inner aggregation query would operate with the $n^{\text{th}} - 1$ row data (usually NULL for row 1) when it should have been using the n^{th} row data.

This problem has existed in various forms for all previous versions of Oracle Rdb7, but only appears in Release 6.1 and later when the inner of the nested queries contains an UPDATE statement.

The following example demonstrates the problem:

```
SQL> ATTACH 'FILENAME SHIPPING';
SQL> SELECT * FROM MANIFEST WHERE VOYAGE_NUM = 4904 OR
cont>      VOYAGE_NUM = 4909;
  VOYAGE_NUM      EXP_NUM  MATERIAL      TONNAGE
  -----
      4904          311    CEDAR          1200
      4904          311    FIR             690
      4909          291    IRON ORE       3000
      4909          350    BAUXITE        1100
      4909          350    COPPER         1200
      4909          355    MANGANESE       550
      4909          355    TIN            500
7 rows selected

SQL> BEGIN
cont> FOR :A AS EACH ROW OF
cont> SELECT * FROM VOYAGE V WHERE V.SHIP_NAME = 'SANDRA C.' OR
cont>      V.SHIP_NAME = 'DAFFODIL' DO
cont>   FOR :B AS EACH ROW OF TABLE CURSOR MODCUR1 FOR
cont>   SELECT * FROM MANIFEST M WHERE M.VOYAGE_NUM = :A.VOYAGE_NUM DO
cont>   UPDATE MANIFEST
cont>     SET TONNAGE = (SELECT (AVG (M1.EXP_NUM) *3) FROM MANIFEST M1
cont>                   WHERE M1.VOYAGE_NUM = :A.VOYAGE_NUM)
cont>     WHERE CURRENT OF MODCUR1;
cont>   END FOR;
cont> END FOR;
cont> END;
SQL> SELECT * FROM MANIFEST WHERE VOYAGE_NUM = 4904 OR
cont>      VOYAGE_NUM = 4909;
  VOYAGE_NUM      EXP_NUM  MATERIAL      TONNAGE
  -----
      4904          311    CEDAR          NULL
      4904          311    FIR            NULL
      4909          291    IRON ORE       933
      4909          350    BAUXITE        933
      4909          350    COPPER         933
      4909          355    MANGANESE       933
      4909          355    TIN            933
7 rows selected
```

The correct value for TONNAGE on both rows for VOYAGE_NUM 4904 (outer query row 1) is $AVG(311+311)*3=933$. However, Oracle Rdb7 calculates it as $AVG(NULL+NULL)*3=NULL$. In addition, the TONNAGE value for VOYAGE_NUM 4909 (outer query row 2) is actually the TONNAGE value for outer query row 1.

A workaround is to declare a variable of the same type as the outer reference data item, assign the outer reference data into the variable before the inner query that contains the correlated aggregation subquery, and reference the variable in the aggregation subquery. Keep in mind the restriction on the use of local variables in FOR cursor loops.

For example:

```

SQL> DECLARE :VN INTEGER;
SQL> BEGIN
cont> FOR :A AS EACH ROW OF
cont> SELECT * FROM VOYAGE V WHERE V.SHIP_NAME = 'SANDRA C.' DO
cont> SET :VN = :A.VOYAGE_NUM;
cont> FOR :B AS EACH ROW OF TABLE CURSOR MODCUR1 FOR
cont> SELECT * FROM MANIFEST M WHERE M.VOYAGE_NUM = :A.VOYAGE_NUM DO
cont> UPDATE MANIFEST
cont> SET TONNAGE = (SELECT (AVG (M1.EXP_NUM) *3) FROM MANIFEST M1
cont> WHERE M1.VOYAGE_NUM = :VN)
cont> WHERE CURRENT OF MODCUR1;
cont> END FOR;
cont> END FOR;
cont> END;
SQL> SELECT * FROM MANIFEST WHERE VOYAGE_NUM = 4904;
  VOYAGE_NUM      EXP_NUM  MATERIAL          TONNAGE
         4904           311    CEDAR              933
         4904           311    FIR                  933

```

This problem was corrected in Oracle Rdb7 Release 7.0.0.2. An updated release note stating that this was fixed was inadvertently left out of all the following sets of release notes. Please note that this issue is now corrected.

6.0.58 Considerations when Using Holdable Cursors

If your applications use holdable cursors, be aware that after a COMMIT or ROLLBACK statement is executed, the result set selected by the cursor may not remain stable. That is, rows may be inserted, updated, and deleted by other users because no locks are held on the rows selected by the holdable cursor after a commit or rollback occurs. Moreover, depending on the access strategy, rows not yet fetched may change before Oracle Rdb actually fetches them.

As a result, you may see the following anomalies when using holdable cursors in a concurrent user environment:

- If the access strategy forces Oracle Rdb to take a data snapshot, the data read and cached may be inaccurate by the time the cursor fetches the data. For example, user 1 opens a cursor and commits the transaction. User 2 deletes rows read by user 1 (this is possible because the read locks are released). It is possible for user 1 to report data now deleted and committed.
- If the access strategy uses indexes that allow duplicates, updates to the duplicates chain may cause rows to be skipped, or even revisited.

Oracle Rdb keeps track of the dbkey in the duplicate chain pointing to the data that was fetched. However, the duplicates chain could be revised by the time Oracle Rdb returns to using it.

Holdable cursors are a very powerful feature for read-only or predominantly read-only environments. However, in concurrent update environments, the instability of the cursor may not be acceptable. The stability of holdable cursors for update environments will be addressed in future versions of Oracle Rdb.

You can define the logical name RDMSS\$BIND_HOLD_CURSOR_SNAP or configuration parameter RDB_BIND_HOLD_CURSOR_SNAP to the value 1 to force all hold cursors to fetch the result set into a cached data area. (The cached data area appears as a "Temporary Relation" in the optimizer strategy displayed by the SET FLAGS STRATEGY statement or the RDMSS\$DEBUG_FLAGS S flag.) This logical name or configuration parameter helps to stabilize the cursor to some degree.

6.0.59 INCLUDE SQLDA2 Statement Is Not Supported for SQL Precompiler for PL/I in Oracle Rdb Release 5.0 or Higher

The SQL statement INCLUDE SQLDA2 is not supported for use with the PL/I precompiler in Oracle Rdb Release 5.0 or higher.

There is no workaround. This problem will be fixed in a future version of Oracle Rdb.

6.0.60 SQL Pascal Precompiler Processes ARRAY OF RECORD Declarations Incorrectly

The Pascal precompiler for SQL gives an incorrect %SQL-I-UNMATEND error when it parses a declaration of an array of records. The precompiler does not associate the END statement with the record definition, and the resulting confusion in host variable scoping causes a fatal error.

A workaround for the problem is to declare the record as a type and then define your array of that type. For example:

```
main.spa:
    program main (input,output);
    type
    exec sql include 'bad_def.pin';    !gives error
    exec sql include 'good_def.pin';  !ok
    var
        a : char;
    begin
    end.

-----
bad_def.pin
x_record = record
y : char;
variable_a: array [1..50] of record
    a_fld1 : char;
    b_fld2 : record;
                t : record
                    v : integer;
            end;
    end;
end;

-----
good_def.pin
good_rec = record
    a_fld1 : char;
    b_fld2 : record
        t : record
            v: integer;
        end;
    end;
end;

x_record = record
y : char
variable_a : array [1..50] of good_rec;
end;
```

6.0.61 RMU Parallel Backup Command Not Supported for Use with SLS

The RMU Parallel Backup command is not supported for use with the Storage Library System (SLS) for OpenVMS.

6.0.62 Oracle RMU Commands Pause During Tape Rewind

Digital UNIX Systems

For Oracle Rdb Release 6.1 or higher on Digital UNIX, the Oracle RMU Backup and Restore commands pause under certain conditions.

If multiple tape drives are used for RMU Backup or RMU Restore commands and a tape needs to rewind, the Oracle RMU command pauses until the rewind is complete. This is different from behavior on OpenVMS systems where the command continues to write to tape drives that are not rewinding.

There is no workaround for this problem.

6.0.63 TA90 and TA92 Tape Drives Are Not Supported on Digital UNIX

Digital UNIX Systems

When rewinding or unloading tapes using either TA90 and TA92 drives, Digital UNIX intermittently returns an EIO error causing the Oracle RMU operation to abort. This problem occurs most often when Oracle RMU accesses multiple tape drives in parallel. However, the problem occurs even with single-tape drive access.

As a result of this problem, Oracle Rdb on Digital UNIX supports neither TA90 nor TA92 tape drives.

6.1 Oracle CDD/Repository Restrictions

This section describes known problems and restrictions in Oracle CDD/Repository Release 7.0 and earlier.

6.1.1 Oracle CDD/Repository Compatibility with Oracle Rdb Features

Some Oracle Rdb features are not fully supported by all versions of Oracle CDD/Repository. Table 6–1 shows which versions of Oracle CDD/Repository support Oracle Rdb features and the extent of support.

In Table 6–1, repository support for Oracle Rdb7 features can vary as follows:

- **Explicit support**—The repository recognizes and integrates the feature, and you can use the repository to manipulate the item.
- **Implicit support**—The repository recognizes and integrates the feature, but you cannot use any repository interface to manipulate the item.
- **Pass-through support**—The repository does not recognize or integrate the feature, but allows the Oracle Rdb7 operation to complete without aborting or overwriting metadata. With pass-through support, a CDD-I-MBLRSYNINFO informational message may be returned.

Table 6–1 Oracle CDD/Repository Compatibility for Oracle Rdb Features

Oracle Rdb Feature	Minimum Release of Oracle Rdb	Minimum Release of Oracle CDD/Repository	Support
CASE, NULLIF, and COALESCE expressions	6.0	6.1	Implicit
CAST function	4.1	7.0	Explicit
Character data types to support character sets	4.2	6.1	Implicit
Collating sequences	3.1	6.1	Explicit
Constraints (PRIMARY KEY, UNIQUE, NOT NULL, CHECK, FOREIGN KEY)	3.1	5.2	Explicit
CURRENT_DATE, CURRENT_TIME, and CURRENT_TIMESTAMP functions	4.1	7.0	Explicit
CURRENT_USER, SESSION_USER, SYSTEM_USER functions	6.0	7.0	Explicit
Date arithmetic	4.1	6.1	Pass-through
DATE ANSI, TIME, TIMESTAMP, and INTERVAL data types	4.1	6.1	Explicit
Delimited identifiers	4.2	6.1 ¹	Explicit
External functions	6.0	6.1	Pass-through
External procedures	7.0	6.1	Pass-through
EXTRACT, CHAR_LENGTH, and OCTET_LENGTH functions	4.1	6.1	Explicit
GRANT/REVOKE privileges	4.0	5.0 accepts but does not store information	Pass-through
Indexes	1.0	5.2	Explicit
INTEGRATE DOMAIN	6.1	6.1	Explicit
INTEGRATE TABLE	6.1	6.1	Explicit
Logical area thresholds for storage maps and indexes	4.1	5.2	Pass-through
Multinational character set	3.1	4.0	Explicit
Multiversion environment (multiple Rdb versions)	4.1	5.1	Explicit
NULL keyword	2.2	7.0	Explicit
Oracle7 compatibility functions, such as CONCAT, CONVERT, DECODE, and SYSDATE	7.0	7.0	Explicit
Outer joins, derived tables	6.0	7.0	Pass-through
Query outlines	6.0	6.1	Pass-through
Storage map definitions correctly restored	3.0	5.1	Explicit

¹The repository does not preserve the distinction between uppercase and lowercase identifiers. If you use delimited identifiers with Oracle Rdb, the repository ensures that the record definition does not include objects with names that are duplicates except for case.

(continued on next page)

Table 6–1 (Cont.) Oracle CDD/Repository Compatibility for Oracle Rdb Features

Oracle Rdb Feature	Minimum Release of Oracle Rdb	Minimum Release of Oracle CDD/Repository	Support
Stored functions	7.0	6.1	Pass-through
Stored procedures	6.0	6.1	Pass-through
SUBSTRING function	4.0	7.0 supports all features 5.0 supports all but 4.2 MIA features ²	Explicit
Temporary tables	7.0	6.1	Pass-through
Triggers	3.1	5.2	Pass-through
TRUNCATE TABLE	7.0	6.1	Pass-through
TRIM and POSITION functions	6.1	7.0	Explicit
UPPER, LOWER, TRANSLATE functions	4.2	7.0	Explicit
USER function	2.2	7.0	Explicit

²Multivendor Integration Architecture (MIA) features include the CHAR_LENGTH clause and the TRANSLATE function.

6.1.2 Multischema Databases and CDD/Repository

You cannot use multischema databases with CDD/Repository and Oracle Rdb release 7.0 and earlier. This problem will be corrected in a future release of Oracle Rdb.

6.1.3 Interaction of Oracle CDD/Repository Release 5.1 and Oracle RMU Privileges Access Control Lists

Oracle Rdb provides special Oracle RMU privileges that use the unused portion of the OpenVMS access control list (ACL) to manage access to Oracle RMU operations.

You can use the RMU Set Privilege and RMU Show Privilege commands to set and show the Oracle RMU privileges. The DCL SHOW ACL and DIRECTORY/ACL commands also show the added access control information; however, these tools cannot translate the names defined by Oracle Rdb.

Note

The RMU Convert command propagates the database internal ACL to the root file for access control entries (ACEs) that possess the SECURITY and DBADM (ADMINISTRATOR) privileges.

Oracle CDD/Repository protects its repository (dictionary) by placing the CDD\$SYSTEM rights identifier on each file created within the anchor directory. CDD\$SYSTEM is a special, reserved rights identifier created by Oracle CDD/Repository.

When Oracle CDD/Repository executes the DEFINE REPOSITORY command, it adds (or augments) an OpenVMS default ACL to the anchor directory. Typically, this ACL allows access to the repository files for CDD\$SYSTEM and denies access to everyone else. All files created in the anchor directory inherit this default ACL, including the repository database.

Unfortunately, there is an interaction between the default ACL placed on the repository database by Oracle CDD/Repository and the Oracle RMU privileges ACL processing.

Within the ACL on the repository database, the default access control entries (ACEs) that were inherited from the anchor directory will precede the ACEs added by RMU Restore. As a result, the CDD\$SYSTEM identifier will not have any Oracle RMU privileges granted to it. Without these privileges, if the user does not have the OpenVMS SYSPRV privilege enabled, Oracle RMU operations, such as Convert and Restore, will not be allowed on the repository database.

The following problems may be observed by users who do not have the SYSPRV privilege enabled:

- While executing a CDO DEFINE REPOSITORY or DEFINE DICTIONARY command:
 - If the CDD\$TEMPLATEDB backup (.rbf) file was created by a previous version of Oracle Rdb7, the automatic RMU Convert operation that will be carried out on the .rbf file will fail because SYSPRV privilege is required.
 - If the CDD\$TEMPLATEDB backup (.rbf) file was created by the current version of Oracle Rdb7, the restore of the repository database will fail because the default ACEs that already existed on the repository file that was backed up will take precedence, preventing RMU\$CONVERT and RMU\$RESTORE privileges from being granted to CDD\$SYSTEM or the user.
 - If no CDD\$TEMPLATEDB is available, the repository database will be created without a template, inheriting the default ACL from the parent directory. The ACE containing all the required Oracle RMU privileges will be added to the end of the ACL; however, the preexisting default ACEs will prevent any Oracle RMU privilege from being granted.
- You must use the RMU Convert command to upgrade the database disk format to Oracle Rdb7 after installing Release 7.0. This operation requires the SYSPRV privilege.

During the conversion, RMU Convert adds the ACE containing the Oracle RMU privileges at the end of the ACL. Because the repository database already has the default Oracle CDD/Repository ACL associated with it, the Oracle CDD/Repository ACL will take precedence, preventing the granting of the Oracle RMU privileges.
- During a CDO MOVE REPOSITORY command, the Oracle RMU privilege checking may prevent the move, as the RMU\$COPY privilege has not been granted on the repository database.
- When you execute the CDD template builder CDD_BUILD_TEMPLATE, the step involving RMU Backup privilege has not been granted.

Oracle CDD/Repository Releases 5.2 and higher correct this problem. A version of the Oracle CDD/Repository software that corrects this problem and allows new repositories to be created using Oracle Rdb7 is provided on the Oracle Rdb7 kit for use on OpenVMS VAX systems. See Section 6.1.3.1 for details.

6.1.3.1 Installing the Corrected CDDSHR Images

OpenVMS VAX Systems

Note

The following procedure must be carried out if you have installed or plan to install Oracle Rdb7 and have already installed CDD/Repository Release 5.1 software on your system.

Due to the enhanced security checking associated with Oracle RMU commands in Oracle Rdb on OpenVMS VAX, existing CDDSHR images for CDD/Repository Release 5.1 must be upgraded to ensure that the correct Oracle RMU privileges are applied to newly created or copied repository databases.

Included in the Oracle Rdb7 for OpenVMS VAX distribution kit is a CDD upgraded image kit, called CDDRDB042, that must be installed after you have installed the Oracle Rdb7 for OpenVMS VAX kit.

This upgrade kit should be installed by using VMSINSTAL. It automatically checks which version of CDDSHR you have installed and replaces the existing CDDSHR.EXE with the corrected image file. The existing CDDSHR.EXE will be renamed SYSSLIBRARY:OLD_CDDSHR.EXE.

The upgrade installation will also place a new CDD_BUILD_TEMPLATE.COM procedure in SYSSLIBRARY for use with CDD/Repository V5.1.

Note

If you upgrade your repository to CDD/Repository V5.1 after you install Oracle Rdb7 V7.0, you must install the corrected CDDSHR image again to ensure that the correct CDDSHR images have been made available.

The CDD/Repository upgrade kit determines which version of CDD/Repository is installed and replaces the existing CDDSHR.EXE with the appropriate version of the corrected image.

6.1.3.2 CDD Conversion Procedure

OpenVMS VAX Systems

Oracle Rdb7 provides RDB\$CONVERT_CDD\$DATABASE.COM, a command procedure that both corrects the anchor directory ACL and performs the RMU Convert operation. The command procedure is located in SYSSLIBRARY.

Note

You must have SYSPRV enabled before you execute the procedure RDB\$CONVERT_CDD\$DATABASE.COM because the procedure performs an RMU Convert operation.

Use the procedure RDB\$CONVERT_CDD\$DATABASE.COM to process the anchor directory and update the ACLs for both the directory and, if available, the repository database.

This procedure accepts one parameter: the name of the anchor directory that contains, or will contain, the repository files. For example:

```
$ @SYS$LIBRARY:DECRDB$CONVERT_CDD$DATABASE [PROJECT.CDD_REP]
```

If many repositories exist on a system, you may want to create a DCL command procedure to locate them, set the Oracle RMU privileges ACL, and convert the databases. Use DCL commands similar to the following:

```
$ LOOP:
$   REP_SPEC = F$SEARCH("[000000...]CDD$DATABASE.RDB")
$   IF REP_SPEC .NES. ""
$   THEN
$       @SYS$LIBRARY:DECRDB$CONVERT_CDD$DATABASE -
$           'F$PARSE(REP_SPEC,,,"DIRECTORY")'
$       GOTO LOOP
$   ENDIF
```


7.1 Enhancements Provided in Oracle Rdb7 Release 7.0.6

7.1.1 RMU Show Statistics "Hot Standby Throughput" Screen

Monitoring and analyzing Hot Standby throughput is a very difficult task. There are a large number of factors that impact the maximum throughput that Hot Standby is theoretically capable of supporting. Unfortunately, there are no RMU Show Statistic screens that directly provide this type of information, although the statistic information is available. Furthermore, Hot Standby throughput varies over time, as the application workload changes and other outside influences affect network bandwidth and capacity.

This problem has been corrected by an enhancement in Oracle Rdb7 Release 7.0.6. The RMU Show Statistic utility contains a new "Hot Standby Throughput" screen that resides in the "Hot Standby Information" sub-menu. The purpose of this screen is to provide information about the capacity, both theoretical and actual, of the Hot Standby product, as well as how well the current throughput relates to the throughput capacity.

The "Hot Standby Throughput" screen contains two regions: an informational region that displays the theoretical throughput, the computed capacity and the actual throughput; and a circular chart that displays the current throughput as a percentage of the computed capacity over time.

In the informational region, three lines are displayed. The first line displays the maximum throughput rate ever achieved during the collection period, plus a computed theoretical maximum. The second line displays the running average rate during the collection period, plus a computed throughput capacity. Finally, the third line displays the current rate, and its percentage of the computed capacity.

The following is an example of the "Hot Standby Throughput" screen:

```

Node: ALPHA3 (1/1/1) Oracle Rdb X7.0-00 Perf. Monitor 17-MAR-2000 15:22:53.75
Rate: 0.50 Seconds Hot Standby Throughput Elapsed: 00:01:17.71
Page: 1 of 1 KODH$:[R_ANDERSON.TMP]TCS.RDB;1 Mode: Online

```

```

-----
Max I/O:      8 Max Blocks:    427 Theoretical:    3416
Avg I/O:      2 Avg Blocks:    104 Capacity:        416
Cur I/O:     3 Cur Blocks:    303 Throughput Percent: 100.0
-----
91-100% | 0 | 30 3 0000 | 00 | 0 | 00 0 0000 | 0 0 0
81-90%  | 0 | 3 | 00 | 0 | 0 | 0 | 9
71-80%  |  |  |  |  | 7 |  |
61-70%  |  |  |  |  |  |  | 9 | 7
51-60%  |  |  |  |  | 5 |  |  |
-----
41-50%  |  | 4 |  |  | 6 |  |  | 5 | 0 9
31-40%  |  |  |  |  | 9 |  |  |  |  |
21-30%  |  |  |  |  |  |  |  |  |  |
11-20%  | 1 8 |  | 0 | 7 |  | 6 6
1-10%   | 4 | 4 | 0 | 8 7 | 4 | 9 |
-----
15:22:20 15:22:26 15:22:31 15:22:37 15:22:43 15:22:49 15:22:15
Sample interval is 0.50 Seconds
-----

```

```

Exit Help Menu Reset Set_rate Unreset Write !

```

Ideally, the current throughput percentage should be as close as possible to 100% of the throughput capacity. However, application run-time issues, network saturation, standby database apply rates, etc., all factor into the equation. Also, the database synchronization mode plays a large part in the Hot Standby throughput. Therefore, the normal expected behaviour is in the 51% to 100% region of the chart. Throughput above the 50% level indicates acceptable throughput, while throughput consistently less than 50% may be indicative of a resource problem.

The maximum capacity is computed based on the running average of the real-time Hot Standby throughput. Therefore, it is normal to consistently have 100% throughput during the initial startup period, until such time as throughput achieves its steady state.

The "Hot Standby Throughput" screen information is not recorded in the binary output file. Therefore, the screen is not available during binary file replay.

7.1.2 RMU/UNLOAD/AFTER_JOURNAL /STATISTICS Totals Information

When using the "STATISTICS" qualifier with the RMU/UNLOAD/AFTER_JOURNAL command, the number of records extracted for each table being unloaded is displayed.

This information has been enhanced in this release, 7.0.6, to include the total number of records for all tables being extracted.

The following example includes the "Total" information as the last line of the statistics output:

```

$ RMU /UNLOAD /AFTER /STATISTICS MFP MFP.AIJBCK -
  /TABLE=(NAME=C1,OUT=C1) -
  /TABLE=(NAME=C2,OUT=C2) -
  /TABLE=(NAME=C3,OUT=C3) -
  /TABLE=(NAME=C4,OUT=C4)
-----
ELAPSED: 0 00:00:00.29 CPU: 0:00:00.04 BUFIO: 18 DIRIO: 14 FAULTS: 307
Table "C1" : 2 records written (2 modify, 0 delete)
Table "C2" : 4 records written (4 modify, 0 delete)
Table "C3" : 1 record written (1 modify, 0 delete)
Table "C4" : 6 records written (6 modify, 0 delete)
Total : 13 records written (13 modify, 0 delete)

```

7.1.3 RMU/SHOW STATISTIC "Row Cache Information" Cache Name Shortcut

When using the RMU/SHOW STATISTIC utility to display Row Cache information, the "=" shortcut character can be used to quickly select a different row cache. This avoids having to return to the current sub-menu and re-selecting the same screen, which is the normal method for selecting a specific row cache.

7.1.4 RMU Command Line Help Updated

RMU command line help has been enhanced in Oracle Rdb7 Release 7.0.6 to include help for the following features:

- LogMiner for Rdb; specifically:
 - RMU Set Logminer command
 - RMU Unload After_Journal command
 - Using LogMiner for Rdb
- Hot Standby; specifically:
 - RMU Replicate After_Journal commands

7.1.5 New /EXCLUDE_TABLES Qualifier for RMU/COLLECT OPTIMIZER

Bug 1156806

For Oracle Rdb7 Release 7.0.6, a new optional "/EXCLUDE_TABLES=(table_list)" qualifier has been added to the RMU/COLLECT OPTIMIZER command to allow the user to specify a list of database tables to be excluded from statistics collection and update for statistics used by the Rdb query optimizer.

This optional qualifier cannot be negated and must be specified with a value which is a list of one or more database tables to be excluded from the statistics collection and update. If the /TABLES and /EXCLUDE_TABLES qualifiers are both used in the same RMU/COLLECT OPTIMIZER command, the /EXCLUDE_TABLES qualifier takes precedence so that if the same table is specified in both the /TABLES and /EXCLUDE_TABLES lists, that table will be excluded from the statistics collection and update.

The following example uses the new option.

```
RMU/COLLECT OPTIMIZER MF_PERSONNEL /LOG/EXCLUDE_TABLES=(EMPLOYEES,DEPARTMENTS)
```

7.1.6 RMU Show Statistic Utility "Hot Standby Log" Facility

The RMU Show Statistic Utility has been enhanced to provide a "Hot Standby Log" facility. The purpose of this facility is to provide a tabular recording of the "Hot Standby Statistics" screen.

Since a large portion of the "Hot Standby Statistics" screen is determined at runtime, this information is not stored in the binary output file. The "Hot Standby Log" facility allows you to capture this runtime information in a human-readable format.

The "Hot Standby Log" facility is enabled using two different methods. These are:

1. The `/HOT_STANDBY_LOG` qualifier can be used to specify the name of the desired Hot Standby logfile; for example: `/HOT_STANDBY_LOG=HS.LOG`
2. The "Start hot standby logging" option of the Tools menu ("!") can be used to specify the name of the desired Hot Standby logfile at runtime.

The "Hot Standby Log" facility can be disabled at runtime using the "Stop hot standby logging" option of the Tools menu ("!").

The following is an example of the "Hot Standby Log" output:

```
Oracle Rdb X7.0-00 Performance Monitor Hot Standby Log
Database KODH$:[R_ANDERSON.TMP]TCS.RDB;1
Hot Standby Log created 20-JUN-2000 07:32:51.18
Cur.Dat/Tim HS.State   User Auto DB.Lag.Time Cur.MSN  Cur.AIJ.Pos  Alt.AIJ.Pos
07:33:37.83 DB Synch.   Cold Cold 00:00:00.00      1 1:2
07:33:46.50 TCP/IP      Cold Cold 00:00:00.00      3 1:2          1:2
07:33:46.80 TCP/IP      Cold Cold 00:00:00.00      4 1:3          1:2
```

The following columns of information are displayed:

- **Cur.Dat/Time**
This is the current time the log entry was recorded.
- **HS.State**
This is the current Hot Standby state.
- **User**
This is the user-specified synchronization mode.
- **Auto**
This is the current database synchronization mode. It may vary from the user-specified mode if the replication governor is enabled.
- **DB.Lag.Time**
This is the time by which the standby database lags behind the master database.
- **Cur.MSN**
This is the current message sequence number being sent from the master database, or being processed on the standby database.
- **Cur.AIJ.Pos**
This is the current database AIJ position, expressed as a sequence number and a block number.
- **Alt.AIJ.Pos**

This is the alternate database AIJ position, expressed as a sequence number and a block number. If the master database is being logged, this value represents the standby database AIJ position.

Duplicate entries are not recorded. Therefore, the output logfile may not contain entries for each refresh interval.

The “Hot Standby Log” is created even if Hot Standby is inactive, but it is written *only* when Hot Standby is active.

7.1.7 Impact of NUMBER OF RECOVERY BUFFERS on Row Cache DBR Performance

The database “NUMBER OF RECOVERY BUFFERS” parameter specifies the number of database buffers that the recovery (DBR) process will use while performing recovery operations. As with user processes, the number of buffers for a DBR process can have a dramatic effect on performance. Generally, increasing the number of buffers for the DBR process will have a positive effect on performance when the DBR process is performing recovery for large transactions; either for rollback of aborted transactions or for recovery (REDO) of many or large transactions when the database FAST COMMIT feature is enabled.

The performance of the database recovery (DBR) process can be especially significant when the Row Cache feature is enabled. After a database or node failure (system crash, for example) for a database with Row Cache enabled, when the database is re-opened, the monitor will create a DBR process to recover the database. This DBR process performs the following steps:

1. All row caches are recovered from the backing store (.RDC) files
2. The oldest checkpoint (of either the "fast commit" or the Record Cache Server (RCS) checkpoint) for any database user is determined
3. All transactions starting at the oldest checkpoint found are (re)applied to the database
4. Each active transaction is rolled back

Because of the potential database I/O required to perform steps 3 and 4, a larger number of buffers can help reduce the duration of the database recovery process.

Testing demonstrates how significant the number of DBR buffers can be on the performance of re-opening a database after node failure. A test case involving 25 users performing 1,000 transactions each (a total of 25,000 transactions) was interrupted by a simulated system crash. After the system was restarted, the database was opened. A specially instrumented database recovery (DBR) process was used to measure the amount of CPU time consumed along with the number of I/Os performed for various portions of the recovery.

With the default of 20 recovery buffers for the DBR process, a total of 53,542 disk I/Os were performed during the REDO portion of recovery. At a rate of 100 I/Os per second, this step would require about 9 minutes. Increasing the buffer count to 1,000 reduced the number of disk I/Os for this step to 4,342. At the same rate of 100 I/Os per second, the REDO step would now take less than 1 minute. However, for this particular test, increasing the number of buffers for the DBR process to 2,000 only reduced the I/O count to 4,262 and further increases of the buffer count resulted in no additional I/O decrease.

Generally, if global buffers are not enabled and there is sufficient memory on the system, a large number of recovery buffers for the DBR process is beneficial. It is important, however, to avoid specifying so many buffers that the DBR process page faults excessively.

When global buffers are enabled, the number of buffers for the database recovery process is limited to the global buffer USER LIMIT quota. In order to increase the number of buffers for the DBR process, both the "USER LIMIT" and "NUMBER OF RECOVERY BUFFERS" parameters may need to be increased.

For node failure recovery when the ROW CACHE feature is enabled, a value of 5000 or 10000 for buffers for the DBR process may be reasonable. The optimum number of buffers will vary greatly depending on the number and complexity of transactions and processes to be recovered and is thus quite application and site specific.

7.1.8 RMU Show Statistics "Row Cache Overview" Screen

The RMU Show Statistic utility has been enhanced in Oracle Rdb7 Release 7.0.6 to provide a new "Row Cache Overview" screen. The purpose of this screen is to provide an overview of cache information for all row caches on a single screen.

The following is an example of the "Row Cache Overview" screen:

```

Node: MONGO (1/1/1)      Oracle Rdb X7.0-00 Perf. Monitor 28-AUG-2000 09:42:52.58
Rate: 3.00 Seconds      Row Cache Overview (Unsorted)      Elapsed: 00:08:41.45
Page: 1 of 1            ROOT_DEVICE:[RDB_64WH]TPCC.RDB;1      Mode: Online
-----
Cache.Name..... #Searches  Hit% Full%  #Inserts #Wrap   #Slots  Len
STOCK              1278227  72.8  75.5    348025  0     460800  320
STOCK_HASH         1275201  81.6  92.5    234365  0     253440  100
STOCK_SYS          1275201  81.6  92.5    234365  0     253440   16
NEW_ORDER_SORT     526601   99.3  22.7     4113   0      18000  476
ORDER_LINE_SORT    3410785  99.4  97.8    19539  1     15360  964
ORDER_SORT         490600   99.1  83.9     4317   0      5120   476
WAREHOUSE           89087   99.9 100.0      64   1         64  116
WAREHOUSE_HASH     89099   99.9  25.0     16   0         64   80
WAREHOUSE_SYS      89099   99.9  25.0     16   0         64   16
DISTRICT           93617   99.6  50.0     320   0         640  116
DISTRICT_HASH      93224   99.8  25.0     160   0         640   92
DISTRICT_SYS       93225   99.8  25.0     160   0         640   16
ITEM                451404   88.1  53.4    53398  0    100010   96
ITEM_HASH           451976   96.8  98.9    14146  0     14300  148
ITEM_SYS            451986   96.8  98.9    14146  0     14300   16
CUSTOMER_SORT      235711   92.6  54.3    17385  0     32000  964
RDB$SYSTEM          20846   92.7 100.0     1488  1         960   432
-----
Config Exit Help Menu >next_page <prev_page Options Pause Reset Set_rate Write

```

The "Row Cache Overview" screen displays the following information:

- **Cache.Name**
This column displays the name of a row cache mapped by the process executing the RMU Show Statistic utility.
- **#Searches**
This column identifies the total number of cache searches that have been performed for a particular row cache.
- **Hit%**
This column identifies the cache hit percentage for a particular row cache.
- **Full%**

This column identifies the fullness percentage for a particular row cache.

- #Inserts

This column identifies the total number of cache inserts that have been performed for a particular row cache.

- #Wrap

This column identifies the total number of cache insert cursor wraps for a particular row cache.

- #Slots

This column identifies the total number of slots for a particular row cache.

- Len

This column identifies the length of each slot for a particular row cache.

Using the “Config” on-screen menu option, the following screen configuration options are available:

- Display mapped row caches

This option indicates that only row caches currently mapped by the RMU Show Statistic utility should be displayed. This is the default mapping option.

- Display all row caches

This option indicates that all valid row caches should be displayed. This means the RMU Show Statistic utility will map all known row caches. This may take a while to complete.

- Unsorted display

This option indicates that the display should not be sorted in any manner. This is the default sort option.

- Sort by number of searches

This option indicates that the display should be sorted in descending order by total number of cache searches.

- Sort by hit percentage

This option indicates that the display should be sorted in descending order by cache hit percentage.

- Sort by fullness percentage

This option indicates that the display should be sorted in descending order by cache fullness percentage.

- Sort by number of inserts

This option indicates that the display should be sorted in descending order by total number of cache inserts.

- Sort by row cache name

This option indicates that the display should be sorted in ascending alphabetical row cache name order.

- Sort by number of wraps

This option indicates that the display should be sorted in descending order by total number of cache insert cursor wraps.

- Sort by number of slots

This option indicates that the display should be sorted in descending order by cache slot count.

- Sort by slot size

This option indicates that the display should be sorted in descending order by cache slot size.

This screen is not available during binary file replay.

7.1.9 SHOW STATS "Row cache (one field)" Display Configuration

The RMU Show Statistic utility "Row cache (one field)" screen has been enhanced to allow user-defined sort configurations.

Using the "Config" on-screen menu, the following sort options are available:

- Unsorted display.
This is the default option, and displays the row cache information in an unsorted sequence.
- Alphabetical.
This option sorts the display in ascending row cache name sequence.
- Max rate per second.
This option sorts the display in descending sequence, based on the maximum rate per second.
- Average rate per second.
This option sorts the display in descending sequence, based on the average rate per second.
- Total count.
This option sorts the display in descending sequence, based on the total count.
- Average per transaction.
This option sorts the display in descending sequence, based on the average per transaction.

The screen name in the header region reflects the selected sort option. The following is an example of the row cache screen, sorted in name sequence.


```

-----
statistic..... rate.per.second..... total..... average.....
name..... max..... cur..... avg..... count..... per.trans....
I_TSR_PA_RPT0          0          0          0.0          0          0.0
I_TSR_PA_RPT1          0          0          0.0          0          0.0
I_TSR_PA_RPT10         0          0          0.0          0          0.0
I_TSR_PA_RPT11         0          0          0.0          0          0.0
I_TSR_PA_RPT12         0          0          0.0          0          0.0
I_TSR_PA_RPT13         0          0          0.0          0          0.0
I_TSR_PA_RPT14         0          0          0.0          0          0.0
I_TSR_PA_RPT15         0          0          0.0          0          0.0
I_TSR_PA_RPT16         0          0          0.0          0          0.0
I_TSR_PA_RPT17         0          0          0.0          0          0.0
I_TSR_PA_RPT18         0          0          0.0          0          0.0
I_TSR_PA_RPT19         0          0          0.0          0          0.0
I_TSR_PA_RPT2          0          0          0.0          0          0.0
I_TSR_PA_RPT3          0          0          0.0          0          0.0
I_TSR_PA_RPT4          0          0          0.0          0          0.0
-----

```

Config Exit Graph Help Menu >next_page <prev_page Options Reset Set_rate Time_pl

7.1.10 LogMiner Now Supports SQL*Loader Output Files

LogMiner now has the capability to produce output files that can be used by SQL*Loader to load the extracted data into an Oracle database. This feature requires fixed text format for the data file and that the CONTROL table option be used instead of the RECORD_DEFINITION table option. The CONTROL table option can be specified on either the command line or in an options file.

The following example shows the new syntax:

```

$ RMU/UNLOAD/AFTER TEST_DB TEST_DB_AIJ1_BCK -
  /FORMAT=TEXT -
  /TABLE=(NAME=TEST_TBL, -
          OUTPUT=LOGMINER_TEXT.TXT, -
          CONTROL=LOGMINER_CONTROL.CTL, -
          TABLE_DEFINITION=TEST_TBL.SQL)

```

The following is an example of a control file produced by LogMiner. The control file is specific to a fixed length record text file. NULLs are handled by using the NULLIF clause for the column definition which references a corresponding null byte filler column. There is a null byte filler column for each column in the underlying table but not for the LogMiner specific columns at the beginning of the record. If a column is NULL, the corresponding RDBSLM_NBn filler column will be set to "1". VARCHAR columns are padded with blanks but the blanks will be ignored by default when loading with SQL*Loader. If you wish to preserve the blanks, you can update the control file and add the "PRESERVE BLANKS" clause.

```

-- Control file for LogMiner transaction data 25-AUG-2000 12:15:50.47
-- From database table "TEST_DB"
LOAD DATA
INFILE 'DISK:[DIRECTORY]LOGMINER_TEXT.TXT;'
APPEND INTO TABLE 'RDB_LM_TEST_TBL'
(
RDB$LM_ACTION                POSITION(1:1) CHAR,
RDB$LM_RELATION_NAME         POSITION(2:32) CHAR,
RDB$LM_RECORD_TYPE           POSITION(33:44) INTEGER EXTERNAL,
RDB$LM_DATA_LEN              POSITION(45:50) INTEGER EXTERNAL,
RDB$LM_NBV_LEN               POSITION(51:56) INTEGER EXTERNAL,
RDB$LM_DBK                   POSITION(57:76) INTEGER EXTERNAL,
RDB$LM_START_TAD             POSITION(77:90) DATE "YYYYMMDDHHMISS",
RDB$LM_COMMIT_TAD            POSITION(91:104) DATE "YYYYMMDDHHMISS",
RDB$LM_TSN                   POSITION(105:124) INTEGER EXTERNAL,
RDB$LM_RECORD_VERSION        POSITION(125:130) INTEGER EXTERNAL,
TEST_COL                     POSITION(131:150) CHAR NULLIF RDB$LM_NB1 = 1,
RDB$LM_NB1                   FILLER POSITION(151:151) INTEGER EXTERNAL
)

```

This capability has been added in Oracle Rdb7 Release 7.0.6.

7.1.11 RMU /UNLOAD /AFTER_JOURNAL "/FORMAT" Qualifier

\ LMDELIMITEDTEXT)

The RMU /UNLOAD /AFTER_JOURNAL command now supports the qualifier "/FORMAT". Similar to the keyword on the standard RMU /UNLOAD command, /FORMAT controls how the output data will be formatted.

The "/FORMAT" qualifier options are:

- Format=(Binary)

If you specify the Format=Binary option, Oracle RMU outputs data to a fixed-length binary flat file. This is the default output format.

- Format=(Text)

If you specify the Format=Text option, Oracle RMU converts all data to printable text before unloading it. VARCHAR(n) strings are padded with blanks when the specified string has fewer characters than n so that the resulting string is n characters long.

- Format=(Delimited_Text [,delimiter-options])

If you specify the Format=Delimited_Text option, Oracle RMU applies delimiters to all data before unloading it. Note that DATE VMS dates are output in the collatable time format, which is yyyyymmddhhmmsscc. For example, March 20, 1993 is output as: 1993032000000000.

Delimiter options (and their default values if you do not specify delimiter options) are:

- Prefix=string

Specifies a prefix string that begins any column value in the ASCII output file. If you omit this option, the column prefix will be a quotation mark (").

- Separator=string

Specifies a string that separates column values of a row. If you omit this option, the column separator will be a single comma (,).

- Suffix=string

Specifies a suffix string that ends any column value in the ASCII output file. If you omit this option, the column suffix will be a quotation mark (").

- Terminator=string

Specifies the row terminator that completes all the column values corresponding to a row. If you omit this option, the row terminator will be the end of the line.

- Null=string

Specifies a string, which when found in the database column, is unloaded as NULL in the output file. The Null option can be specified on the command line as any one of the following:

- A quoted string
- An empty set of double quotes ("")
- No string

The string that represents the null character must be quoted on the Oracle RMU command line. You cannot specify a blank space or spaces as the null character. You cannot use the same character for the Null value and other Delimited_Text options.

7.1.12 New WORKLOAD Item for RMU/EXTRACT

In Oracle Rdb7 Release 7.0.6, RMU Extract allows the work load and statistics stored in the RDB\$WORKLOAD table to be unloaded as a script of RMU/INSERT OPTIMIZER_STATISTICS statement. This unloaded information can be applied after a new database is created using the SQL EXPORT and IMPORT statements, or it can be applied to a similar database for use by RMU/COLLECT OPTIMIZER_STATISTICS/STATISTIC=WORKLOAD.

A new WORKLOAD keyword has been added to the /ITEM qualifier. The default is /ITEM=NOWORKLOAD. The WORKLOAD item is not included in the ALL item.

This item generates a DCL command language script for OpenVMS as shown in the example below. The script includes references to Rdb system tables as well as user tables.

```

$ RMU/EXTRACT/ITEM=WORKLOAD -
  SCRATCH/LOG/OPTION=(FILENAME,AUDIT)
$! RMU/EXTRACT for Oracle Rdb X7.0-00 7-SEP-2000 22:00:42.72
$!
$! WORKLOAD Procedure
$!
$!-----
$ SET VERIFY
$ SET NOON
$
$! Created on 7-SEP-2000 10:12:26.36
$! Last collected on 7-SEP-2000 22:00:34.47
$!
$ RMU/INSERT OPTIMIZER_STATISTICS -
  SCRATCH -
  /TABLE=(CUSTOMERS) -
  /COLUMN_GROUP=(CUSTOMER_NAME) -
  /DUPLICITY_FACTOR=(4.0000000) -
  /NULL_FACTOR=(0.0000000) /LOG
$
$! Created on 7-SEP-2000 10:12:26.36
$! Last collected on 7-SEP-2000 22:00:34.58
$!
$ RMU/INSERT OPTIMIZER_STATISTICS -
  SCRATCH -
  /TABLE=(RDB$FIELDS) -
  /COLUMN_GROUP=(RDB$FIELD_NAME) -
  /DUPLICITY_FACTOR=(1.7794118) -
  /NULL_FACTOR=(0.0000000) /LOG
$
.
.
.
$ SET NOVERIFY
$ EXIT

```

The following options can be used to modify the output of the WORKLOAD item.

- /OPTION=AUDIT_COMMENT
Each RMU/INSERT OPTIMIZER_STATISTICS statement is preceded by the created and altered date for the workload entry. The default is /OPTION=NOAUDIT_COMMENT.
- /OPTION=FILENAME_ONLY
The database file specification output for the RMU/INSERT OPTIMIZER_STATISTICS statement is abbreviated to just the filename portion.

7.1.13 RMU /UNLOAD /AFTER_JOURNAL “/INCLUDE = ACTION” Qualifier

The RMU /UNLOAD /AFTER_JOURNAL command now supports the qualifier “/INCLUDE = ACTION”. This qualifier accepts the keywords “[NO]DELETE” and “[NO]MODIFY” to control if deleted or modified records are to be extracted from the after-image journal.

The “/INCLUDE = ACTION” qualifier options are:

- [NO]DELETE
If you specify DELETE, pre-deletion record contents will be extracted from the AIJ. If you specify NODELETE, then no pre-deletion record contents will be extracted. The default is DELETE.
- [NO]MODIFY

If you specify MODIFY, modified or added record contents will be extracted from the AIJ. If you specify NOMODIFY, then no modified or added record contents will be extracted. The default is MODIFY.

The default is /INCLUDE=ACTION=(DELETE, MODIFY) where both pre-deletion and modify/add record contents are extracted.

7.1.14 SHOW STATS "Logical Area Access" Logfile

The RMU/SHOW STATISTIC utility now provides the ability to record "access and modification" of application relations (tables) and, optionally, Rdb system relations. When the "Logical Area Access" facility is enabled, all modifications of tables are recorded in a human-readable logfile.

The purpose of the "Logical Area Access" facility is to provide a mechanism for recording which tables are modified over a period of time. The record of access includes the table name as well as the number of fetches, inserts, updates and erases to that table since the epoch.

Note

The "Logical Area Access" facility is not an audit mechanism and identification of which user/process accessed the tables is not provided.

The new qualifier /ACCESS_LOG identifies the name of the logfile where logical area accesses are to be recorded. The configuration variable ACCESS_LOG can also be used to identify the logfile name.

The configuration variable SYSTEM_LOGICAL_AREAS can be used to specify that only application logical areas are to be monitored. The default value "TRUE" indicates that all tables, including system tables, are to be monitored. The value "FALSE" indicates that only application (non-system) logical areas are to be monitored. There is no corresponding command qualifier for this configuration variable.

The configuration variable LOGICAL_AREA_FILTER can be used to specify a filter to prune the logical areas to be monitored. The filter can be a regular expression. The filter pattern string may contain either one or both of the two wildcard characters asterisk (*) and percent (%). The asterisk character is mapped to zero or more characters. The percent character is mapped to only one character.

The command qualifier /REOPEN_INTERVAL can be specified to identify the threshold, in seconds, after which a new version of the logfile will be automatically created. This allows you to view logfiles recorded over extended periods of time. The configuration variable REOPEN_INTERVAL can also be used for this purpose.

The following is an example of a logical area access logfile, using a refresh rate of one second, which is quite extreme. Ordinarily, a refresh rate of one minute (60 seconds) is recommended.

Oracle Rdb X7.0-00 Performance Monitor Logical Area Access Log
 Database KODH\$:[R_ANDERSON.TMP]TCS.RDB;6
 Logical Area Access Log created 8-SEP-2000 14:00:18.27

CurrentTime	Fetch	Store	Update	Erase	Logical.Area.Name
14:08:55.97	10002	650	0	0	TRAN_SUMMARY_RECON4
14:08:56.64	10002	657	0	0	TRAN_SUMMARY_RECON4
14:08:57.72	0	4640	0	0	TRAN_SUMMARY_RECON0
14:08:57.72	0	2170	0	0	TRAN_SUMMARY_RECON1
14:08:57.72	0	160	0	0	TRAN_SUMMARY_RECON11
14:08:57.72	10002	810	0	0	TRAN_SUMMARY_RECON2
14:08:57.72	0	961	0	0	TRAN_SUMMARY_RECON3
14:08:57.72	0	50	0	0	TRAN_SUMMARY_RECON13
14:08:57.72	10002	660	0	0	TRAN_SUMMARY_RECON4
14:08:57.72	0	1280	0	0	TRAN_SUMMARY_RECON9
14:08:58.82	0	4650	0	0	TRAN_SUMMARY_RECON0
14:08:58.82	10002	820	0	0	TRAN_SUMMARY_RECON2
14:08:58.82	0	1286	0	0	TRAN_SUMMARY_RECON9
14:08:58.82	0	20	0	0	TRAN_SUMMARY_RECON19
14:08:59.90	0	4690	0	0	TRAN_SUMMARY_RECON0
14:08:59.91	0	1290	0	0	TRAN_SUMMARY_RECON9
14:08:59.91	0	30	0	0	TRAN_SUMMARY_RECON19
14:09:00.97	0	4700	0	0	TRAN_SUMMARY_RECON0
14:09:00.97	0	40	0	0	TRAN_SUMMARY_RECON19
14:09:02.03	0	4730	0	0	TRAN_SUMMARY_RECON0
14:09:02.04	0	60	0	0	TRAN_SUMMARY_RECON19
14:09:03.12	0	4750	0	0	TRAN_SUMMARY_RECON0
14:09:03.12	0	90	0	0	TRAN_SUMMARY_RECON19
14:09:04.20	0	4760	0	0	TRAN_SUMMARY_RECON0
14:09:04.20	0	830	0	0	TRAN_SUMMARY_RECON5

The logical area access logfile contains six columns of information. These are:

- **CurrentTime**
This is the time when the entry was recorded.
- **Fetch**
This is the current number of row fetches performed against the designated table since the epoch.
- **Store**
This is the current number of row inserts performed against the designated table since the epoch.
- **Update**
This is the current number of row updates performed against the designated table since the epoch.
- **Erase**
This is the current number of row deletions performed against the designated table since the epoch.
- **Logical.Area.Name**
This is the name of the logical area.

7.1.15 New OPTIMIZE Qualifier for RMU Unload

With Oracle Rdb7 Release 7.0.6, a new qualifier has been added to RMU Unload to allow the user to control the query optimization of the unload command.

The OPTIMIZE qualifier provides the following options:

- TOTAL_TIME

This option requests that total time optimization be applied to the unload query. It does not override the setting if any query outline is used.

In some cases total time optimization may improve performance of the RMU Unload command when the query optimizer favors overall performance instead of faster retrieval of the first row. Since RMU Unload is unloading the entire set, there is no need to require fast delivery of the first few rows.

This option may not be specified at the same time as FAST_FIRST. TOTAL_TIME is now the default for RMU Unload.

- FAST_FIRST

This is the inverse of TOTAL_TIME and was the default for prior versions of RMU Unload. This option asks the query optimizer to favor strategies that return the first rows quickly, possibly at the expense of longer overall retrieval time. This option will not override the setting if any query outline is used.

Note

Oracle does not recommend this optimization preference for RMU Unload. It is provided only for backward compatibility with prior Rdb releases.

This option may not be specified at the same time as TOTAL_TIME.

- SEQUENTIAL_ACCESS

This option requests that index access be disabled for this query. This is particularly useful for RMU Unload from views against strictly partitioned tables. Strict partitioning is enabled by the PARTITIONING IS NOT UPDATABLE clause on the CREATE or ALTER STORAGE MAP statements. Retrieval queries will only use this type of partition optimization during sequential table access.

This option may not be specified at the same time as USING_OUTLINE.

- USING_OUTLINE = outline_name

This option supplies the name of the query outline to be used by RMU Unload. If the query outline does not exist then the name will be ignored.

This option may not be specified at the same time as SEQUENTIAL_ACCESS.

- NAME_AS = query_name

This option supplies the name of the query. It is used to annotate output from the Rdb debug flags (enabled using the logical RDMSSSET_FLAGS) and is also logged by Oracle TRACE.

If USING_OUTLINE is not used, then this name will also be used as the query outline name.

- CONFORMANCE = { OPTIONAL | MANDATORY }

This option accepts two keywords, `OPTIONAL` or `MANDATORY`, which can be used to override the settings in the specified query outline.

If the matching query outline is invalid, then conformance of `MANDATORY` will cause the query compile, and hence the RMU Unload operation, to stop. The query outline will be one which either matches the string provided by the `USING_OUTLINE` or `NAME_AS` options, or matches the query identification.

The default behavior is to use the setting within the query outline. If no query outline is found, or query outline usage is disabled, then this option is ignored.

The following example shows the output from the flags `STRATEGY` and `ITEM_LIST` which indicate that the `/OPTIMIZE` call requested that sequential access be used, and also that `TOTAL_TIME` is used as the default optimizer preference.

```
$ DEFINE RDMS$SET_FLAGS "STRATEGY,ITEM_LIST"
$ RMU/UNLOAD/OPTIMIZE=SEQUENTIAL_ACCESS PERSONNEL EMPLOYEES E.DAT
.
.
.
~H Request Information Item List: (len=11)
0000 (00000) RDB$K_SET_REQ_OPT_PREF "0"
0005 (00005) RDB$K_SET_REQ_OPT_SEQ "1"
000A (00010) RDB$K_INFO_END
Get      Retrieval sequentially of relation EMPLOYEES
%RMU-I-DATRECUNL, 100 data records unloaded.
```

7.1.16 RCS Can Ignore Duplicate Checkpoint Requests

The Record Cache Server (RCS) process accepts and queues global checkpoint requests and then processes them in the order received. If a large number of checkpoint requests are generated, the RCS process can spend a significant amount of time performing the checkpoints.

An optional feature of the RCS is to ignore duplicate no-wait global checkpoint requests. If there is already a queued no-wait global checkpoint request, new checkpoint requests that are also for no-wait global checkpoints are ignored. This prevents the RCS from accumulating many additional checkpoint requests. Note that an existing executing checkpoint request is not considered queued; it is already executing. So when the RCS is ignoring duplicate requests, there can be at most one no-wait global checkpoint request queued for execution and another request currently executing.

In general, this optional feature does not need to be enabled. Oracle suggests that you only enable this feature if you experience a large number of global checkpoint requests.

To enable this feature, define a system-wide logical name `RDM$BIND_RCS_IGNORE_DUPLICATE_CHECKPOINT_REQUEST` to "1" prior to opening the database(s).

7.1.17 RMU /UNLOAD /AFTER_JOURNAL "/PARAMETER" Qualifier

The RMU `/UNLOAD /AFTER_JOURNAL` command now supports the qualifier `"/PARAMETER"`. This qualifier accepts one or more quoted strings that are concatenated together and passed to any callback routines. The `"/PARAMETER"` qualifier allows the user-supplied callback routines to be passed arguments from the command line.

For each table being extracted, an initial call to the callback routine is made with the "Action" field set to "P" to indicate command line parameter passing. The contents of the "/PARAMETER" qualifier strings are passed in the data portion of the record.

7.1.18 New /CLOSE_WAIT Qualifier for RMU/RESTORE Command

Bug 1228099

For Oracle Rdb7 Release 7.0.6, a new /CLOSE_WAIT qualifier has been added to the RMU/RESTORE command.

- /CLOSE_WAIT=n

This qualifier cannot be negated and must be specified with a value which is the number of minutes to wait before the database is automatically closed. This qualifier can only be specified if /OPEN_MODE=AUTOMATIC has also been specified. It cannot be used if /OPEN_MODE=MANUAL is specified. The /CLOSE_WAIT qualifier specifies a database close mode of TIMED AUTOMATIC. Note that the already existing /OPEN_MODE qualifier also sets the database close mode.

The following example uses the new qualifier to set the database close mode to TIMED AUTOMATIC, specifying that the database will be closed automatically in 10 minutes. The /OPEN_MODE must be set to AUTOMATIC. If /CLOSE_WAIT were not specified, the close mode would be set to the specified open mode, which is AUTOMATIC.

```
RMU/RESTORE/OPEN_MODE=AUTOMATIC/CLOSE_WAIT=10/DIR=DISK:[DIRECTORY] TEST_DB.RBF
RMU/DUMP/HEADER=PARAMETERS TEST_DB.RDB
```

7.1.19 New /EXCLUDE_TABLES Qualifier for RMU/COLLECT OPTIMIZER_STATISTICS

Bug 1156806

For Oracle Rdb7 Release 7.0.6, a new /EXCLUDE_TABLES qualifier has been added to the RMU/COLLECT OPTIMIZER_STATISTICS command.

- /EXCLUDE_TABLES=(table_1,table_2,table_3,...)

This qualifier cannot be negated and must be specified with a value which is a list of one or more tables to be excluded from the statistics collection. If the /TABLES and /EXCLUDE_TABLES qualifiers are both used in the same RMU/COLLECT OPTIMIZER command the /EXCLUDE_TABLES qualifier takes precedence. Therefore, if a table is specified in both the /TABLES and /EXCLUDE_TABLES lists, that table will be excluded from the statistics collection.

The following example uses the new /EXCLUDE_TABLES qualifier and shows that it takes precedence over the /TABLES qualifier.

```
RMU/COLLECT OPTIMIZER /LOG/TABLES=(DEGREES,EMPLOYEES,JOBS)-
/EXCLUDE_TABLES=(DEGREES,EMPLOYEES) MF_PERSONNEL
```

Optimizer Statistics collected for table : JOBS

```
Cardinality          : 15
Row clustering factor : 8.9333334

Workload Column group : JOB_CODE
Duplicity factor      : 1.0000000
Null factor           : 0.0000000
```

```

Workload Column group : WAGE_CLASS
Duplicity factor      : 3.7500000
Null factor           : 0.0000000
Done writing stats.... at 19-OCT-2000 09:54:45.21

```

7.1.20 CREATE MODULE Now Supports External Routines

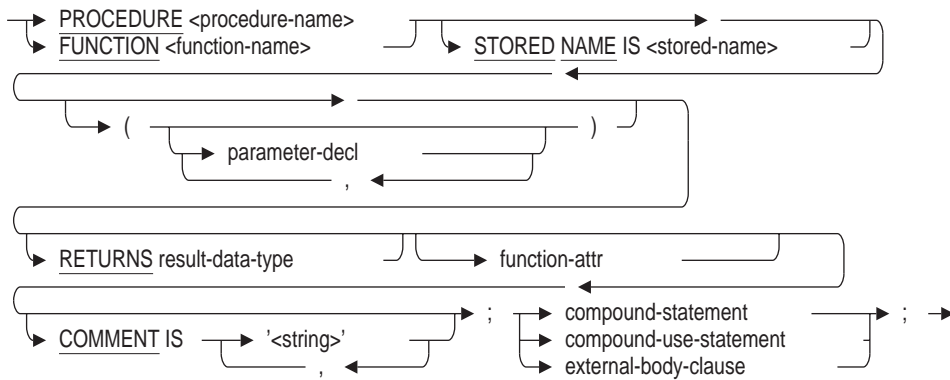
Bugs 914442 and 754171

7.1.20.1 DESCRIPTION

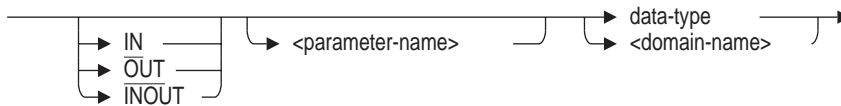
In prior versions of Oracle Rdb, the CREATE MODULE statement allowed only SQL stored functions and procedures to be defined.

With this release of Rdb7, syntax for external functions and procedures is now permitted within a CREATE MODULE statement. Both SQL and external routine definitions can be mixed in a single module. The revised syntax for the <routine-clause> is shown below. Please refer to the Oracle Rdb7 SQL Reference Manual for information on the CREATE MODULE syntax, and the <routine-clause> of CREATE FUNCTION and CREATE PROCEDURE which is described in the CREATE ROUTINE section.

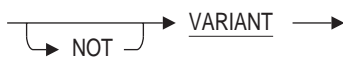
routine-clause =



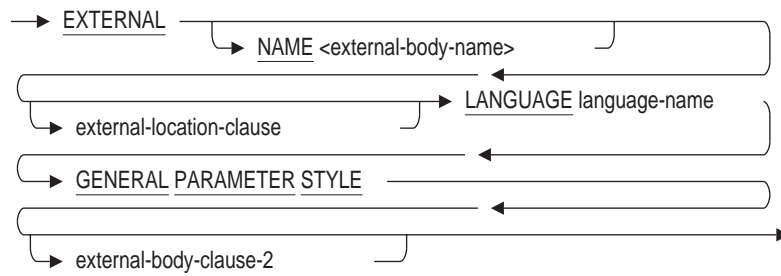
parameter-decl =



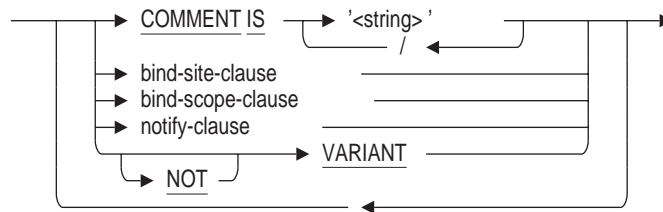
function-attr =



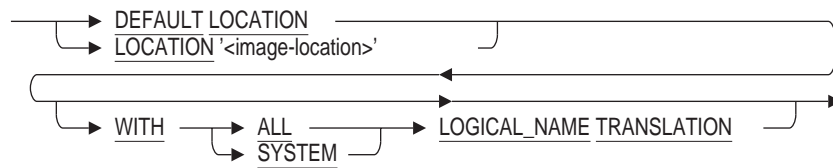
external-body-clause =



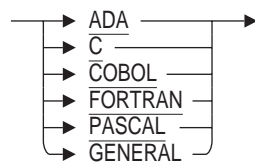
external-body-clause-2 =



external-location-clause =



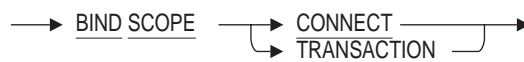
language-name =



bind-site-clause =

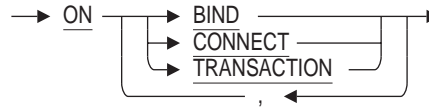


bind-scope-clause =



notify-clause =

→ NOTIFY notify-entry-name



This change provides the following benefits:

- Both stored and external routines can be collected within a single module. This allows simplified DROP, GRANT and REVOKE operations which will operate on multiple external routines in a single statement. For instance, a DROP MODULE can be used to remove external and stored routines in a single command. GRANT and REVOKE can be applied to a larger set of routines, rather than requiring individual statements for each external routine.
- Related routines, whether external or stored, can be grouped together thus providing simplified database maintenance.
- External routines within the same module now share the same database environment. This allows, for instance, one external routine to OPEN a cursor, another to FETCH rows and another to CLOSE the cursor.

In contrast, when an external routine is created using the CREATE FUNCTION or CREATE PROCEDURE syntax, only that routine uses the database environment.

7.1.20.2 USAGE NOTES

- The name of the stored module need not be the same as that used for the SQL module language module, or SQL pre-compiled module which implements the external functionality. However, keeping identical or similar names may assist future maintenance of the application.
- The shared module database environment is only significant for those external routines which execute SQL statements.
- If an external routine attaches to a database, it will be implicitly disconnected when the invoking session is terminated.

However, Oracle recommends that the current transaction, open cursors and session started for the external function be terminated before using DISCONNECT. This can be done explicitly by calling an external routine which terminates the transaction and disconnects in the same context as the invoking routine, or it can be done implicitly when using a NOTIFY routine.

7.1.20.3 EXAMPLE

Using External Cursors

This example uses multiple external routines to manage a table cursor in the external routine database environment. This management includes the OPEN, FETCH and CLOSE of a single cursor.

Several domains are defined so that parameter data types can be consistently defined in the database that contains the application and also the database upon which the cursor is open.

```

create domain SQLSTATE_T char(5);
create domain STATUS_CODE char(1);
create domain STATUS_NAME char(8);
create domain STATUS_TYPE char(14);

```

The external function interface is contained within a single CREATE MODULE statement. This module also contains the application in a single stored SQL procedure.

```

create module EX
  language SQL

  -- These procedures define the interface to the external
  -- routines that implement the transaction and cursor operations
  --
  procedure EX_START_READ_TXN
    (inout :ss sqlstate_t);
    external location 'TEST$SCRATCH:EX.EXE'
    language general
    general parameter style
    comment is 'start a READ ONLY transaction';

  procedure EX_COMMIT
    (inout :ss sqlstate_t);
    external location 'TEST$SCRATCH:EX.EXE'
    language general
    general parameter style;

  procedure EX_OPEN_CURSOR
    (inout :ss sqlstate_t);
    external location 'TEST$SCRATCH:EX.EXE'
    language general
    general parameter style
    comment is 'find all rows in WORK_STATUS order by STATUS_CODE';

  procedure EX_CLOSE_CURSOR
    (inout :ss sqlstate_t);
    external location 'TEST$SCRATCH:EX.EXE'
    language general
    general parameter style;

  procedure EX_FETCH_CURSOR
    (inout :ss sqlstate_t,
     out :s_code STATUS_CODE, out :s_code_ind integer,
     out :s_name STATUS_NAME, out :s_name_ind integer,
     out :s_type STATUS_TYPE, out :s_type_ind integer);
    external location 'TEST$SCRATCH:EX.EXE'
    language general
    general parameter style;

  -- This SQL procedure implements a simple application
  --
  procedure WORK_STATUS
    comment is 'Use an external cursor to fetch all rows in the'
    /      'WORK_STATUS table';
    begin
    declare :s_code STATUS_CODE;
    declare :s_name STATUS_NAME;
    declare :s_type STATUS_TYPE;
    declare :s_code_ind, :s_name_ind, :s_type_ind integer;
    declare :ss sqlstate_t;

    -- start a read-only transaction on the PERSONNEL database
    call EX_START_READ_TXN (:ss);
    if :ss ^= '00000' then
      SIGNAL :ss;
    end if;

```

```

-- open the cursor on the work-status table
call EX_OPEN_CURSOR (:ss);
if :ss ^= '00000' then
    SIGNAL :ss;
end if;

-- now loop and fetch all the rows
FETCH_LOOP:
loop
    call EX_FETCH_CURSOR (:ss,
                          :s_code, :s_code_ind,
                          :s_name, :s_name_ind,
                          :s_type, :s_type_ind);

    case :ss
    when '02000' then
        -- no more rows to fetch
        leave FETCH_LOOP;

    when '00000' then
        begin
            -- we have successfully fetched a row, so display it
            trace 'Status Code: ', case when :s_code_ind < 0
                                         then 'NULL'
                                         else :s_code
                                         end;

            trace 'Status Name: ', case when :s_name_ind < 0
                                         then 'NULL'
                                         else :s_name
                                         end;

            trace 'Status Type: ', case when :s_type_ind < 0
                                         then 'NULL'
                                         else :s_type
                                         end;

            trace '***';
        end;
    else
        -- signal will implicitly leave the stored procedure
        SIGNAL :ss;
    end case;
end loop;

-- close the cursor
call EX_CLOSE_CURSOR (:ss);
if :ss ^= '00000' then
    SIGNAL :ss;
end if;

-- commit the transaction
call EX_COMMIT (:ss);
if :ss ^= '00000' then
    SIGNAL :ss;
end if;
end;

end module;

```

The external procedures for this example are written using the SQL module language. However, any language with embedded SQL, such as C, could have been used.

```

module EX
language GENERAL
parameter colons
-- EX: Sample application
-- Process the WORK_STATUS table using a table cursor
--
declare alias filename 'PERSONNEL'

```

```

declare c cursor for
  select status_code, status_name, status_type
  from WORK_STATUS
  order by status_code

procedure EX_START_READ_TXN
  (sqlstate);
begin
  -- abort any stray transactions
  rollback;
  -- start a READ ONLY transaction
  set transaction read only;
end;

procedure EX_COMMIT
  (sqlstate);
  commit work;

procedure EX_ROLLBACK
  (sqlstate);
  rollback work;

procedure EX_OPEN_CURSOR
  (sqlstate);
  open c;

procedure EX_CLOSE_CURSOR
  (sqlstate);
  close c;

procedure EX_FETCH_CURSOR
  (sqlstate,
   :s_code           STATUS_CODE,
   :s_code_ind       integer,
   :s_name           STATUS_NAME,
   :s_name_ind       integer,
   :s_type           STATUS_TYPE,
   :s_type_ind       integer);
  fetch c
  into :s_code indicator :s_code_ind,
       :s_name indicator :s_name_ind,
       :s_type indicator :s_type_ind;

procedure EX_DISCONNECT
  (sqlstate);
  disconnect default;

```

When run, the application calls the external procedures to open the cursor, fetch the rows and display them using the TRACE statement.

```

SQL> set flags 'trace';
SQL>
SQL> call WORK_STATUS ();
~Xt: Status Code: 0
~Xt: Status Name: INACTIVE
~Xt: Status Type: RECORD EXPIRED
~Xt: ***
~Xt: Status Code: 1
~Xt: Status Name: ACTIVE
~Xt: Status Type: FULL TIME
~Xt: ***
~Xt: Status Code: 2
~Xt: Status Name: ACTIVE
~Xt: Status Type: PART TIME
~Xt: ***
SQL>

```

Oracle recommends that the cursors be closed and the external routine's database environment be disconnected before the calling session is disconnected. This can be achieved by using NOTIFY routines.

For example, the external procedure that starts the transaction could be modified as shown below to declare a NOTIFY routine (EX_RUNDOWN) that, when called, would close the cursors, rollback the transaction and disconnect from the database.

```

procedure EX_START_READ_TXN
  (inout :ss sqlstate_t);
external location 'TEST$SCRATCH:EX.EXE'
language general
general parameter style
notify EX_RUNDOWN on BIND
comment is 'start a READ ONLY transaction';

```

The BIND notification ensures that EX_RUNDOWN will be called during the DISCONNECT of the caller and allow the transaction to be rolled back and the session disconnected. ROLLBACK or COMMIT will implicitly close any open cursors, unless the cursors were defined as WITH HOLD. In this case, it is important to also close that cursor. Code similar to the following (in C) could implement this rundown routine.

```

#include <string.h>
#include <stdio.h>
#define RDB$K_RTX_NOTIFY_ACTV_END 2
#define SQLSTATE_LEN 5
void sql_signal ();
void EX_CLOSE_CURSOR (char sqlstate [SQLSTATE_LEN]);
void EX_DISCONNECT (char sqlstate [SQLSTATE_LEN]);
void EX_ROLLBACK (char sqlstate [SQLSTATE_LEN]);

extern void EX_RUNDOWN
  (int *func_code,
   int *u1,           /* U1, U2, U3 are currently unused */
   int *u2,           /* and are reserved for future use */
   int *u3)
{
  char sqlstate [SQLSTATE_LEN];

  if (*func_code == RDB$K_RTX_NOTIFY_ACTV_END)
  {
    /* we are running down this external routine, so
     * close the cursor
     */
    EX_CLOSE_CURSOR (sqlstate);
    if (memcmp ("00000", sqlstate, SQLSTATE_LEN) != 0
        && memcmp ("24000", sqlstate, SQLSTATE_LEN) != 0)
      /* we expect success or maybe 24000 (bad cursor state)
       */
      sql_signal ();

    /* rollback the transaction
     */
    EX_ROLLBACK (sqlstate);
    if (memcmp ("00000", sqlstate, SQLSTATE_LEN) != 0
        && memcmp ("25000", sqlstate, SQLSTATE_LEN) != 0)
      /* we expect success or maybe 25000 (bad transaction state)
       */
      sql_signal ();
  }
}

```



```

        /* disconnect from the database
        */
        EX_DISCONNECT (sqlstate);
        if (memcmp ("00000", sqlstate, SQLSTATE_LEN) != 0)
            /* we expect success or maybe 25000 (bad transaction state)
            */
            sql_signal ();
    }
}

```

The application can be compiled and built using this fragment of DCL code:

```

$      if f$getsyi("arch_name") .eqs. "VAX"
$      then
$      create ex.opt
universal = EX_START_READ_TXN
universal = EX_COMMIT
universal = EX_ROLLBACK
universal = EX_OPEN_CURSOR
universal = EX_CLOSE_CURSOR
universal = EX_FETCH_CURSOR
universal = EX_DISCONNECT
universal = EX_RUNDOWN
psect_attr = RDB$MESSAGE_VECTOR,noshr
psect_attr = RDB$DBHANDLE,noshr
psect_attr = RDB$TRANSACTION_HANDLE,noshr
sql$user/library
$      else
$      create ex.opt
symbol_vector = (EX_START_READ_TXN = procedure)
symbol_vector = (EX_COMMIT = procedure)
symbol_vector = (EX_ROLLBACK = procedure)
symbol_vector = (EX_OPEN_CURSOR = procedure)
symbol_vector = (EX_CLOSE_CURSOR = procedure)
symbol_vector = (EX_FETCH_CURSOR = procedure)
symbol_vector = (EX_DISCONNECT = procedure)
symbol_vector = (EX_RUNDOWN = procedure)
psect_attr = RDB$MESSAGE_VECTOR,noshr
psect_attr = RDB$DBHANDLE,noshr
psect_attr = RDB$TRANSACTION_HANDLE,noshr
sql$user/library
$      endif
$
$      cc EX_RUNDOWN
$      sql$mod EX
$      link/share EX,EX_RUNDOWN,EX/option

```

7.2 Enhancements Provided in Oracle Rdb7 Release 7.0.5

7.2.1 SHOW STATISTIC "Checkpoint Analysis" Screen

The RMU Show Statistic Utility “Online Analysis” facility has been enhanced. The “Checkpoint Analysis” screen performs basic review and analysis of the database checkpoint and process recovery information. The purpose of this screen is to identify processes whose recovery may impact database throughput or availability.

The “Checkpoint Analysis” screen is available even when the “AIJ Fast Commit” feature is not enabled. However, some of the analysis may not be performed in this case.

The following is an example of the “Checkpoint Analysis” screen display:

```
Node: ALPHA3 (1/1/1) Oracle Rdb X7.0-00 Perf. Monitor 13-JAN-2000 08:05:49.34
Rate: 1.00 Second Checkpoint Analysis Elapsed: 5 21:17:06.33
Page: 1 of 1 KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS.RDB;1 Mode: Global
```

```
-----
Process 3C82F330:1 checkpoint 15:2 lags behind current AIJ sequence 18
Process 3C82F330:1 checkpoint 15:2 exceeds 512 block threshold
Process 3C82F330:1 process recovery duration 123 seconds exceeds 10 threshold
Process 3C82F330:1 database freeze duration 123 seconds exceeds 15 threshold
Process 3C82E731:1 checkpoint 17:6586 lags behind current AIJ sequence 18
Process 3C82E731:1 checkpoint 17:6586 exceeds 512 block threshold
Process 3C82E731:1 database freeze duration 6 seconds exceeds 15 threshold
Process 3C82FD33:35 checkpoint 18:2216 exceeds 512 block threshold
Process 3C827752:1 checkpoint 18:2217 exceeds 512 block threshold
Process 3C82DF55:1 checkpoint 18:2142 exceeds 512 block threshold
Process 3C830948:1 checkpoint 18:2218 exceeds 512 block threshold
-----
```

```
Config Exit Help Menu Set_rate Write !
```

The “Checkpoint Analysis” screen performs the following analysis operations:

- **Checkpoint Stale.** This analysis determines whether or not the process checkpoint occurs within the current AIJ journal, which is always desirable.

This analysis results in the following message being displayed:

```
Process 3C82F330:1 checkpoint 15:2 lags behind current AIJ sequence 18
```

- **Checkpoint Old.** This analysis determines whether or not the checkpoint size exceeds a user-specified threshold, expressed in AIJ blocks. The number of AIJ blocks constitutes a physical process recovery duration, but also impacts other components, such as AIJ backup and Row Cache.

The default checkpoint block count threshold is 512 blocks. The default threshold can be modified in the following manner:

- The logical `RDM$BIND_STATS_CHECKPOINT_BLOCK_COUNT` can be defined to specify a different threshold at utility startup.
- The configuration variable `CHECKPOINT_BLOCK_COUNT` can be defined to specify a different threshold in the configuration file.
- The threshold can be modified at run-time using the “Config” on-screen menu option, by selecting the “Checkpoint block count” option.

This analysis results in the following message being displayed:

```
Process 3C82F330:1 checkpoint 15:2 exceeds 512 block threshold
```

- **RUJ File Size.** This analysis determines whether or not the process RUJ file size exceeds a user-specified threshold, expressed in blocks. The number of RUJ blocks constitutes a transaction recovery duration.

The default RUJ file size threshold is 256 blocks. The default threshold can be modified in the following manner:

- The logical `RDM$BIND_STATS_RUJ_FILE_SIZE` can be defined to specify a different threshold at utility startup.
- The configuration variable `RUJ_FILE_SIZE` can be defined to specify a different threshold in the configuration file.
- The threshold can be modified at run-time using the “Config” on-screen menu option, by selecting the “RUJ file size” option.

This analysis results in the following message being displayed:

```
Process 3C82C943:1 RUJ size 30 block exceeds 25 block threshold
```

- **Transaction Rollback Duration.** This analysis determines whether or not the transaction rollback duration exceeds a user-defined threshold, expressed in seconds.

The default transaction rollback threshold is 5 seconds. The default threshold can be modified in the following manner:

- The logical `RDMSBIND_TX_UNDO_DURATION` can be defined to specify a different threshold at utility startup.
- The configuration variable `TX_UNDO_DURATION` can be defined to specify a different threshold in the configuration file.
- The threshold can be modified at run-time using the “Config” on-screen menu option, by selecting the “Transaction rollback duration” option.

This analysis results in the following message being displayed:

```
Process 3C82F330:1 Transaction rollback duration 13 seconds exceeds 10 second threshold
```

- **Process Recovery Duration.** This analysis determines whether or not the recovery of a process failure (transaction REDO) exceeds a user-defined threshold, expressed in seconds.

The default process recovery threshold is 10 seconds. The default threshold can be modified in the following manner:

- The logical `RDMSBIND_TX_REDO_DURATION` can be defined to specify a different threshold at utility startup.
- The configuration variable `TX_REDO_DURATION` can be defined to specify a different threshold in the configuration file.
- The threshold can be modified at run-time using the “Config” on-screen menu option, by selecting the “Process recovery duration” option.

This analysis results in the following message being displayed:

```
Process 3C82F330:1 process recovery duration 123 seconds exceeds 10 second threshold
```

- **Database freeze duration.** This analysis determines whether or not the entire database freeze duration exceeds a user-defined threshold, expressed in seconds. The database freeze duration includes both transaction rollback, process recovery (transaction REDO) and DBR processing.

The default database freeze threshold is 15 seconds. The default threshold can be modified in the following manner:

- The logical `RDMSBIND_DBR_FREEZE_DURATION` can be defined to specify a different threshold at utility startup.
- The configuration variable `TX_DBR_FREEZE_DURATION` can be defined to specify a different threshold in the configuration file.
- The threshold can be modified at run-time using the “Config” on-screen menu option, by selecting the “Database freeze duration” option.

This analysis results in the following message being displayed:

```
Process 3C82F330:1 Database freeze duration 123 seconds exceeds 5 second
threshold
```

The “Checkpoint Analysis” screen information displays run-time information and is not recorded in the binary output file. Consequently, the screen is also not available during replay of a binary input file.

The information displayed represents information for the current node only, even when cluster-wide statistics collection is enabled.

7.2.2 RMU Show Statistic "Online Analysis Logfile" Facility

The RMU Show Statistic utility has been enhanced to provide an “Online Analysis” log file. The online analysis log file provides hard copy of all of the analysis performed by all of the “Online Analysis” facility screens, without having to actually display any of the screens.

The “Online Analysis” log file is enabled using two different methods:

1. The configuration variable `ONLINE_ANALYSIS_LOG` identifies the name of the log file to contain the online analysis information.
2. At run-time, the online analysis log file can be started and stopped using the Tools menu, obtained via the “!” shortcut. Selecting the “Start online analysis logging” option will create the log file, and selecting the “Stop online analysis logging” will terminate the log file.

Note

There is no command qualifier to directly enable the online analysis log file; the configuration file should be used instead via the `/CONFIG` command qualifier.

The following shows an example of the online analysis log file contents:

```
Oracle Rdb X7.0-00 Performance Monitor Online Analysis Log
Database KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS.RDB:1
Online Analysis Log created 15-JAN-2000 07:36:26.69
07:36:34.56 95th %ile transaction duration: 34.4 seconds
07:36:34.56 95th %ile read/write transaction duration: 31.9 seconds
07:36:34.56 95th %ile read-only transaction duration: 503.9 seconds
07:36:34.56 Log server is Automatic
07:36:34.56 AIJ TCS1 device DPA48: same as storage area
07:36:34.56 AIJ TCS2 device DPA48: same as storage area
07:36:34.56 AIJ TCS3 device DPA48: same as storage area
07:36:34.56 AIJ TCS4 device DPA48: same as storage area
07:36:34.56 AIJ TCS5 device DPA48: same as storage area
07:36:34.56 ARB pool exhausted 56 times
07:36:34.56 12.2% synchronous RUJ I/O above 10.0% threshold
07:36:34.56 9.5% RUJ extends above 2.0% threshold
07:36:34.56 Process 3C830F2A:1 checkpoint 11:1075 exceeds 512 block threshold
07:36:34.56 Process 3C830F2A:1 process recovery duration 14 seconds exceeds 10
second threshold
07:36:34.56 Process 3C83112B:1 checkpoint 10:4911 lags behind current AIJ
sequence 11
07:36:34.56 Process 3C83112B:1 checkpoint 10:4911 exceeds 512 block threshold
07:36:34.56 Process 3C83112B:1 process recovery duration 21 seconds exceeds 10
second threshold
07:36:34.56 Process 3C83112B:1 database freeze duration 21 seconds exceeds 15
second threshold
```

```

07:36:34.56 Process 3C831732:31 checkpoint 11:14290 exceeds 512 block threshold
07:36:34.56 Process 3C831D3C:1 checkpoint 11:14213 exceeds 512 block threshold
07:36:34.56 92617.6 process recovery duration above 2.0 second threshold
07:36:34.56 Full database backup has not been performed since 7-JAN-2000
13:46:55.60
07:36:34.56 17.7% page discard rate above 10.0% threshold (avg 0.1 I/Os)
07:36:34.56 100.0% SPAM page fetch rate above 80.0% total fetched threshold
07:36:34.56 217.6% SPAM page fetch rate above 20.0% record stored threshold
07:36:34.56 data TCS extended 1 time total 0 times
07:36:34.56 data TCS async write I/O stalls 1.2 exceeded average 0.3
07:36:34.56 data TCS sync write I/O stalls 15.7 exceeded average 3.0
07:36:42.49 Process 3C82DD5C:1 excessive deadlocks 12 on waiting for page
10:2757 (PR)
07:36:42.49 258.8% duplicate btree fetch above 15.0% threshold
07:36:42.49 51.0% duplicate btree store above 15.0% threshold
07:36:42.49 34.6% duplicate hash btree fetch above 15.0% threshold
07:36:42.49 34.4% duplicate hash index store above 15.0% threshold
07:36:42.49 Row cache is not allowed

```

The online analysis is performed at the specified screen refresh rate. It is possible to generate a considerable number of entries in the online analysis log file. Therefore, it is recommended that the online analysis log file be used in a non-interactive batch job with a reasonable refresh rate of five, ten or thirty seconds.

7.2.3 New OPTIMIZE Clause DML Statements

Oracle Rdb7 Release 7.0.5 adds a new OPTIMIZE FOR SEQUENTIAL ACCESS clause to SELECT, DELETE and UPDATE statements which want to force the use of sequential access. This is particularly valuable for tables which use the strict partitioning functionality.

When a table's storage map has the attribute PARTITIONING IS NOT UPDATABLE, the mapping of data to a storage area is strictly enforced. This is known as **strict partitioning**. When queries on such tables use sequential access, the optimizer can eliminate partitions which do not match the query WHERE restriction rather than scan every partition.

The following example shows a query that deletes selected rows from a specific partition. This table also includes several indices which may be chosen by the optimizer. This new OPTIMIZE clause forces sequential access. In previous releases, a query outline would have to be created for this query. This new clause effectively creates this query outline on-the-fly.

```

SQL> delete from PARTS_LOG
cont> where parts_id between 10000 and 20000
cont> and expiry_date < :purge_date
cont> optimize for sequential access;

```

Please note that all access performed by such queries will be sequential access. Care should be taken that the I/O being used is acceptable by comparing similar queries using index access.

7.2.4 New RMU /RECLAIM Command

Applications that specify the database attach attribute DBKEY SCOPE IS ATTACH can accumulate locked space and locked DBKEYs within the database. If one user is connected to the database in DBKEY SCOPE IS ATTACH mode, all users are forced to operate in this mode, even if they are explicitly connected in TRANSACTION mode. That is, no one reuses dbkeys until the ATTACH session disconnects.

A new RMU /RECLAIM command has been added to allow database keys of deleted rows to be rapidly "cleaned up" in one or more storage areas. The RMU /RECLAIM command reads and updates all pages in a storage area. Where possible, locked lines and locked free space are "released" so that they will be available for later allocation.

The RMU /RECLAIM command runs on-line (does not require exclusive access). However, if there are any users connected to the database in DBKEY SCOPE IS ATTACH mode, the RMU /RECLAIM operation will have greatly reduced effect. In order to release all possible locked space, there should be no users attached to the database in DBKEY SCOPE IS ATTACH mode. Further, to allow database page locked space to be reclaimed, the database session that "owns" the locked space must be detached from the database. This can be accomplished by having each database attach disconnect and reconnect to the database.

Valid qualifiers for the "RMU /RECLAIM" command are:

- /AREA=(listofareas) to indicate the storage areas to be reclaimed. The default is to reclaim all storage areas.
- /LOG to display a log message at the completion of each storage area.

7.2.5 New RMU /SERVER RECORD_CACHE CHECKPOINT Command

A new "RMU /SERVER RECORD_CACHE CHECKPOINT" command has been added to allow a DBA to force the Record Cache Server (RCS) process to checkpoint all modified rows from cache back to the database. This command also accepts the optional qualifiers "/LOG" and "/WAIT".

7.2.6 RCS Cycles TID Value at Checkpoint Completion

When the Oracle Rdb7 Row Cache feature is enabled, the Row Cache Server (RCS) process will cycle through its transaction ID (TID) values as checkpoint or sweep operations that write modified data from the cache to the database complete. This cycling is intended to free locked rows on database pages from records that have been deleted so that the database keys and page space are available to other processes inserting records into the database.

7.2.7 New Option for the GET DIAGNOSTICS Statement/HOT_STANDBY_MODE

For Oracle Rdb7 Release 7.0.5, a new option has been added to the GET DIAGNOSTICS statement (this option was also available in Release 7.0.4 but the release note on it was mistakenly omitted).

- HOT_STANDBY_MODE

This option returns a text string that indicates if this database is participating in a Hot Standby configuration as a master (returns 'MASTER'), or a standby (returns 'STANDBY'), or is not in such a configuration (returns 'NONE').

The result data type is CHAR (31).

The following example uses the new option.

```

SQL> set flags 'trace';
SQL> declare :hsmode char(31);
SQL> begin
cont> get diagnostics :hsmode = HOT_STANDBY_MODE;
cont> trace :hsmode;
cont> end;
~Xt: NONE
SQL>

```

7.2.8 New Option for the GET DIAGNOSTICS Statement/TRANSACTION_CHANGE_ALLOWED

For Oracle Rdb7 Release 7.0.5, a new option has been added to the GET DIAGNOSTICS statement.

- TRANSACTION_CHANGE_ALLOWED

There are many situations where the SQL language programmer would like to start or end a transaction but does not know if a transaction statement (SET TRANSACTION, COMMIT or ROLLBACK) is currently permitted. The transaction statements are not permitted in the following cases:

- During a multi-database or global transaction. In this case the transaction must be coordinated by the client, not a server based procedure.
- When a BEGIN ATOMIC compound statement is in the outer scope.
- When a FOR cursor loop is active in an outer scope.

This option allows the programmer to detect these restricted locations and conditionally execute a COMMIT, ROLLBACK or SET TRANSACTION as needed.

The result data type is INTEGER. If transaction changes are permitted then a value one (1) will be assigned. Otherwise the result will be zero (0).

The following example shows one use of this new option. The called stored procedure ensures that changes to the transaction state are allowed before proceeding with a ROLLBACK.

```

SQL> create module M1
cont>     language SQL
cont>
cont>     procedure ROLLBACK_THE_CHANGE
cont>     comment is 'Perform a ROLLBACK only '
cont>     /         'if it is permitted';
cont>     begin
cont>     declare :txn_change integer;
cont>     get diagnostics
cont>         :txn_change = TRANSACTION_CHANGE_ALLOWED;
cont>     if :txn_change = 1
cont>     then
cont>         trace '...rolling back';
cont>         rollback;
cont>     else
cont>         trace '...skipping rollback';
cont>     end if;
cont>     end;
cont> end module;
SQL>
SQL> create table RT (a integer);
SQL> insert into RT (a) values (1);
1 row inserted
SQL> commit;
SQL>
SQL> set flags 'trace';
SQL>
SQL> begin
cont> call ROLLBACK_THE_CHANGE ();
cont> set transaction read only;
cont> for :x
cont>     as select * from RT
cont> do
cont>     call ROLLBACK_THE_CHANGE ();
cont>     trace :x.a;
cont> end for;
cont> end;
~Xt: ...rolling back
~Xt: ...skipping rollback
~Xt: 1
SQL>

```

7.2.9 New Hot Standby Logicals

Three new logicals have been added to the Hot Standby product:

- **RDM\$BIND_HOT_NETWORK_RETRY_COUNT**
This logical specifies the number of times the Hot Standby product should attempt to re-connect a disconnected network link. The default value is “1”. There is no minimum nor maximum value (the value “0” means do not attempt to re-connect).
- **RDM\$BIND_HOT_NETWORK_RETRY_DELAY**
This logical specifies the number of seconds to wait before attempting to re-connect a disconnected network link, expressed in seconds. The default value is “1” second. There is no minimum nor maximum value (the value “0” means to immediately attempt to re-connect).
- **RDM\$BIND_LRS_BACKUP_AIJ**

This logical specifies that the after-image journals on the standby database should be backed up, instead of merely being initialized, by the AIJ Backup Server (ABS). This logical accepts the following values: “0” indicates that the AIJ files should be initialized (this is the default); the value “1” indicates the AIJ files should be backed up (backup filespecs must have been previously configured).

7.3 Enhancements Provided in Oracle Rdb7 Release 7.0.4

7.3.1 Suggestion To Increase Field Size On RMU SHOW STATISTIC

The RMU/Show Statistic utility, in the menu under “Logical Area Information” sub-menu, “Logical Area Overview (Tables)” option, the “Logical Area Name” is limited to 20 characters. Customers frequently have table names that are larger than 20 characters, or they might have a tablename.areaname, and if this table is partitioned, some of their area files might have the same beginning part of the name with the end being different. It would be nice to have that 20 characters extend out further. Added per customer request.

The following example shows the current display:

```
Node: ALPHA3 (1/1/24)   Oracle Rdb X7.0-00 Perf. Monitor  2-NOV-1999 13:44:58.20
Rate: 1.00 Second      Logical Area Overview (Tables  Elapsed: 14 07:07:01.47
Page: 1 of 5           KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS.RDB;1   Mode: Global
```

```
-----
Logical.Area.Name... record fetch record store record erase      discarded CurTot
RDB$RELATIONS.RDB$SY      24565      0      0      0
RDB$FIELD_VERSIONS.R      223904      0      0      0
RDB$INDICES.RDB$SYST      31495      15      23      0
RDB$INDEX_SEGMENTS.R      31064      45      69      0
RDB$FIELDS.RDB$SYSTE      27114      0      0      0
RDB$RELATION_FIELDS.      22520      0      0      0
RDB$DATABASE.RDB$SYS      1244      0      0      0
RDB$VIEW_RELATIONS.R      0      0      0      0
RDB$CONSTRAINT_RELAT      0      0      0      0
RDB$CONSTRAINTS.RDB$      0      0      0      0
RDB$STORAGE_MAPS.RDB      2056      15      23      0
RDB$STORAGE_MAP_AREA      524      15      23      0
RDB$INTERRELATIONS.R      0      0      0      0
RDB$COLLATIONS.RDB$S      0      0      0      0
RDB$TRIGGERS.RDB$SYS      0      0      0      0
RDB$RELATION_CONSTRA      0      0      0      0
RDB$RELATION_CONSTRA      0      0      0      0
-----
```

```
Config Exit Help Menu >next_page <prev_page Options Pause Reset Set_rate Write
```

There is no workaround to this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.4. By increasing the terminal display width, the RMU/Show Statistic utility will display a larger portion of the logical area name. For example, with the terminal width set to 90 columns, the above screen appears as follows:

```

Node: ALPHA3 (1/1/24)          Oracle Rdb X7.0-00 Perf. Monitor          2-NOV-1999 13:47:15.35
Rate: 1.00 Second             Logical Area Overview (Tables)      Elapsed: 14 07:09:18.62
Page: 1 of 5                  KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS.RDB;1  Mode: Global

```

```

-----
Logical.Area.Name..... record fetch record store record erase   discarded CurTot
RDB$RELATIONS.RDB$SYSTEM          24565          0          0          0          0
RDB$FIELD_VERSIONS.RDB$SYSTEM     223904         0          0          0          0
RDB$INDICES.RDB$SYSTEM            31495         15         23          0          0
RDB$INDEX_SEGMENTS.RDB$SYSTEM     31064         45         69          0          0
RDB$FIELDS.RDB$SYSTEM             27114         0          0          0          0
RDB$RELATION_FIELDS.RDB$SYSTEM     22520         0          0          0          0
RDB$DATABASE.RDB$SYSTEM           1244          0          0          0          0
RDB$VIEW_RELATIONS.RDB$SYSTEM      0            0          0          0          0
RDB$CONSTRAINT_RELATIONS.RDB$S     0            0          0          0          0
RDB$CONSTRAINTS.RDB$SYSTEM        0            0          0          0          0
RDB$STORAGE_MAPS.RDB$SYSTEM       2056         15         23          0          0
RDB$STORAGE_MAP_AREAS.RDB$SYST    524          15         23          0          0
RDB$INTERRELATIONS.RDB$SYSTEM      0            0          0          0          0
RDB$COLLATIONS.RDB$SYSTEM          0            0          0          0          0
RDB$TRIGGERS.RDB$SYSTEM           0            0          0          0          0
RDB$RELATION_CONSTRAINTS.RDB$S     0            0          0          0          0
RDB$RELATION_CONSTRAINT_FLDS.R     0            0          0          0          0
-----

```

```

Config Exit Help Menu >next_page <prev_page Options Pause Reset Set_rate Write Zoom !

```

7.3.2 SHOW STATS "Logical Area Overview" Enhancements

Currently, the RMU Show Statistic Utility "Logical Area Overview" screen can only be sorted in alphabetical order. This is ideal for finding statistic information for a particular logical area, but is less than ideal when the screen is used for performance analysis.

The RMU Show Statistic Utility "Logical Area Overview" screen has been enhanced to provide the ability to sort the display based on any of the displayed column information. Since the user can configure the screen to display any statistic information in any column, this enhancement provides an extremely powerful tool for performance analysis.

Use the "Config" on-screen menu option to display the available sort options.

The following example shows a sample "Logical Area Overview" screen sorted on column one, records fetched:

```

Node: ALPHA3 (1/1/1)   Oracle Rdb X7.0-00 Perf. Monitor   5-NOV-1999 12:55:34.87
Rate: 0.50 Seconds    Logical Area Overview (Tables)   Elapsed: 2 03:00:48.99
Page: 1 of 4         KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS.RDB;1   Mode: Global

```

```

-----
Logical.Area.Name... record fetch record store record erase      discarded CurTot
TRAN_SUMMARY_RECON2      43278      53187      0      0
TRAN_SUMMARY_RECON4      41732      53187      0      0
TRAN_SUMMARY_RECON6      35282      53187      0      0
TRAN_SUMMARY_RECON8      28196      8192       0      0
TRAN_SUMMARY_RECON12     16384      16384      0      0
TRAN_SUMMARY_RECON14     8192       16384      0      0
TRAN_SUMMARY_RECON16     8192       16384      0      0
LANE_MESSAGE_ID          0           0           0      0
STATS_FILE_INDEX         0           0           0      0
SERIAL_KEY_INDEX         0           0           0      0
LANE                      0           0           0      0
XFER_CONTROL             0           0           0      0
SCHEDULE_MASTER          0           0           0      0
POOL_CARD                0           0           0      0
EMPLOYEES                 0      196552     0      0
CSC_RECON                 0           0           0      0
SECURITY_MODULES         0           0           0      0
-----

```

```

Config Exit Help Menu >next_page <prev_page Options Pause Reset Set_rate Write

```

7.3.3 RCS Can Map All Caches at Database Open

By default, when the Oracle Rdb7 Row Cache feature is enabled, the first process to access a cached table or storage area will create and map the associated row cache(s). However, it is possible to cause the RCS process to create and map all defined row caches when the database is opened.

If the system-wide logical name `RDM$BIND_RCS_INITIAL_MAP_ALL_CACHES` is defined to the value "1" when the RCS process starts, the RCS process will create and map all defined row caches for the database.

The RCS process sorts the cache definitions for 32-bit address space usage from largest to smallest before the caches are created. This should help reduce memory fragmentation when using the `SHARED MEMORY IS SYSTEM` option.

7.3.4 Performance Enhancements When Number of Cluster Nodes is 1

Several performance enhancements have been made to Oracle Rdb7. The majority of these improvements are available for databases allowing access from only one node in a cluster (ie, the database is set to `NUMBER OF CLUSTER NODES IS 1`).

In particular, when the database is set to `NUMBER OF CLUSTER NODES IS 1`, various locks and root file write operations have been eliminated. Particularly in environments where this is a mix of read-only and read-write transactions, this can have a significant performance impact.

The logical name `RDM$BIND_AWL_TSNBLK_LOCKING` can be set to "1" to *avoid* several of these performance optimizations if desired.

Additionally, several internal data structures have been further aligned in memory for improved memory access patterns and shorter code sequences on Alpha processors.

7.3.5 New ROW LENGTH Default Calculated for CREATE CACHE

In prior versions of Oracle Rdb, when CREATE CACHE was used to define a logical cache for an existing table or index but the ROW LENGTH IS clause was omitted, the default length used was determined from the Area Inventory Page (AIP) entry for the logical area. Use RMU/DUMP/LAREA=RDB\$AIP to see these length values. This default would sometimes be an inaccurate measure of the row or node size.

With Oracle Rdb7 Release 7.0.4, the defaulting has been changed so that the row length is derived from the Rdb metadata.

- When creating a logical cache, if a table already exists with the same name as the cache, then that table's current row length is calculated. This will account for any table changes which may have been made since the table was created, i.e. a column was added, dropped or altered in data type or size.

In prior releases, the value used from the AIP may have been out-of-date and may have been too small (rows would not be cached) or too large (memory would be wasted).

The default ROW LENGTH does not take into account the compression attributes of the table. Database administrators are encouraged to compare the default chosen with the actual rows on disk because row compression may allow a smaller row length to be used.

Note

If data in the row is not compressible then it is possible that the compression markers added to the row data will cause the length of the stored row to exceed the default ROW LENGTH. Please examine the stored data to see if this is a concern.

If the table is vertically partitioned then this new default will represent the full row size, not the size of individual partitions. Oracle recommends that care be taken to calculate appropriate ROW LENGTH when caching vertically partitioned tables. In previous versions, the length used was for the first matching logical area which didn't necessarily provide a useful length.

System tables do not explicitly use row compression but are stored in an internal abbreviated format. Therefore, Oracle does not recommend using the default ROW LENGTH for Oracle Rdb system tables but rather database administrators should calculate an appropriate value using the existing data in the system table.

- When creating a logical cache, if a SORTED index exists with the same name as the row cache, then the NODE SIZE specified by the CREATE or ALTER INDEX statement will be used for the ROW LENGTH. If none was used then the default size provided by Rdb will be used (typically this is 430 bytes).

In prior releases the value used from the AIP for a SORTED index allowing duplicates was 215 bytes which was too small to cache the index nodes and may have only allowed the duplicate nodes to be cached.

Note

If an index and a table are given the same name, then Oracle Rdb will use the length from the table and not the index. In this case you must

use an explicit ROW LENGTH clause to provide an acceptable length.

- In all other cases, the ROW LENGTH will default to 256. Oracle recommends that you calculate and specify an appropriate ROW LENGTH when creating physical caches or when creating logical caches for hashed index nodes.

These changes will have no affect on existing row cache definitions.

7.3.6 RMU /CHECKPOINT /WAIT /UNTIL

A new qualifier “/UNTIL=date-and-time” has been added to the “RMU /CHECKPOINT /WAIT” command. The UNTIL qualifier specifies the time at which the RMU /CHECKPOINT /WAIT command will stop waiting for the checkpoint and will return an error back to the user.

If you do not specify the UNTIL qualifier, the wait is indefinite.

7.3.7 RMU Extract Supports New AUDIT_COMMENT Option

Oracle Rdb7 Release 7.0.4 adds new functionality to RMU Extract. A new AUDIT_COMMENT option has been added that annotates the extracted objects with the creation and last alter timestamps as well as the username of the creator. The date/time values are displayed using the current settings of SYS\$LANGUAGE and LIB\$DT_FORMAT.

The default is /OPTION=NOAUDIT_COMMENT.

The following example shows an extract from the generated script when the SYS\$LANGUAGE and LIB\$DT_FORMAT are defined. The language and format will default to ENGLISH and the standard OpenVMS format if these logical names are not defined.

```
$ define LIB$DT_FORMAT LIB$DATE_FORMAT_002,LIB$TIME_FORMAT_001
$ define SYS$LANGUAGE french
$ rmu/extract/out=sys$output/item=domain mf_personnel/opt=audit_comment
.
.
.
-- Created on 8 janvier 1998 13:01:31.20
-- Never altered
-- Created by RDB_EXECUTE
--
create domain ADDRESS_DATA_1
  CHAR (25);
  comment on domain ADDRESS_DATA_1 is
    ' Street name';
.
.
.
```

7.3.8 Revised Oracle Rdb for OpenVMS Client Kit

The content of the Oracle Rdb for OpenVMS Client kit has been revised to reflect current software. This release of the client kit contains the following software:

- DBAPack V7.0.1 for the Windows NT on Intel, Windows 95 and Windows 98 platforms
- SQL/Services for Oracle Rdb Version 7.0-4
- SQL/Services Client for Compaq Tru64 UNIX Version 4.0
- SQL/Services Client for Sun Solaris

- Oracle Rdb ODBC Version 2.10.17 (32 bit version)
- Oracle Rdb ODBC for Mac OS
- Oracle Installer Version 3.3.1.2.4 (replaces Version 3.3.1.0.0)

The following software is no longer provided as part of the Oracle Rdb on OpenVMS Client kit.

- DBAPack for the Windows NT on DEC Alpha platform
- DBAPack for the Windows 3.1 platform
- Oracle Enterprise Manager Version 1.3.5
- Personal Oracle7 Version 7.0.3

7.4 Enhancements Provided in Oracle Rdb7 Release 7.0.3.1

7.4.1 Per-Process Monitoring for SHOW STATS

The purpose of the Per-Process Monitoring facility is to provide a powerful drill-down capability to allow the DBA to analyze process-specific information for a single process, class of processes, or all attached database processes. The facility also provides several screens that display a side-by-side comparison of individual process statistic information, such as I/O, transaction, and record statistics. The Per-Process Monitoring facility presents real-time information and, consequently, does not write its information to the binary output file. Therefore, the Per-Process Monitoring facility is not available during the replay of a binary input file.

The Per-Process Monitoring facility is not available if clusterwide statistic collection is active. Conversely, the clusterwide statistic collection facility is not available when the Per-Process Monitoring facility is active.

For a complete description of the various methods by which the Per-Process Monitoring facility can be activated, see the white paper, *RMU SHOW STATISTIC Utility Per-Process Monitoring Facility* on MetaLink.

7.4.2 New DOMAINS Option for RMU/EXTRACT

In Oracle Rdb7 Release 7.0.3.1, RMU Extract adds more control over the way domain definitions are handled. A new `/OPTION=NODOMAINS` can be used to eliminate the domain name from within metadata objects. When used the domain name is replaced by the underlying data type. This option is designed for use with tools that do not understand the SQL92 SQL language domain feature.

- Affect on `/LANGUAGE=SQL` output.

The default is `/OPTION=DOMAINS`.

A SQL script generated when using `/OPTION=NODOMAINS` no longer includes the domain name in the `CREATE TABLE` column definition, `CREATE FUNCTION` or `CREATE PROCEDURE` parameter definitions, and any other value expression which uses the `CAST` function to convert expression to a domain data type (such as a `CREATE VIEW` or `CREATE TRIGGER`).

The output for `CREATE MODULE` is not affected by `/OPTION=NODOMAINS` because it is based on the original source SQL which is not edited by RMU Extract.

- Affect on the `/LANGUAGE=ANSI_SQL` output.

The default is `/OPTION=NODOMAINS` when `/OPTION=NORMAL` is specified or is the default.

In prior versions, ANSI_SQL would attempt to extract the definition using syntax from ANSI SQL89 if /OPTION=NORMAL was specified or was the default. This SQL language standard does not support domain definitions but RMU Extract would still output the definitions if /ITEM=ALL was specified. In this release of Oracle Rdb RMU Extract no longer generates a list of CREATE DOMAIN statements if /ITEM=DOMAINS or /ITEM=ALL is used.

To get the behavior of previous versions use the /OPTION=DOMAINS qualifier to have domains generated. For example:

```
$ RMU/EXTRACT/LANGUAGE=ANSI_SQL/OPTION=DOMAINS databasename
```

Use the /OPTION=FULL qualifier to have syntax generated for SQL92 to include the use of domains.

7.4.3 New NO REORGANIZE clause for ALTER STORAGE MAP

In Oracle Rdb V7.0.2, support was added to allow an existing storage map to be converted to a strictly partitioned storage map using the ALTER STORAGE MAP...PARTITIONING IS NOT UPDATABLE clause of the ALTER STORAGE MAP statement.

This statement implicitly performs a REORGANIZE on the base table, moving rows within the map if necessary, scanning the storage areas to make sure all the stored data conforms to the storage map definition. This allows the the Oracle Rdb optimizer to use this type of table efficiently when a sequential scan uses a subset of the storage areas.

In many cases the database administrator knows that a large table is already strictly partitioned but it is prohibitive to reorganize the table (the amount of I/O alone might last several hours). Therefore, in this release of Oracle Rdb, we have allowed the database administrator to bypass the automatic REORGANIZE performed by the ALTER STORAGE MAP ... PARTITIONING IS NOT UPDATABLE statement using a new NO REORGANIZE clause.

There is an associated risk because Oracle Rdb has not validated the table partitioning. It is possible that rows will be missed by sequential scans. The database administrator must take this risk into account when using this new clause. Oracle Corporation suggests that an ALTER STORAGE MAP...REORGANIZE be carried out as soon as practical.

When the NO REORGANIZE clause is used, Oracle Rdb records this information in the Oracle Rdb system tables. The SHOW STORAGE MAP statement will display informational text.

The following example shows an ALTER STORAGE MAP statement which disabled the area scan for PARTITIONING IS NOT UPDATABLE and the informational text from the SHOW STORAGE MAP statement:

```

SQL> set flags 'stomap_stats';
SQL> alter storage map EMPLOYEES_MAP
cont>     partitioning is not updatable
cont>     no reorganize
cont>     store
cont>     using (EMPLOYEE_ID)
cont>     in EMPIDS_LOW
cont>         with limit of ('00200')
cont>     in EMPIDS_MID
cont>         with limit of ('00400')
cont>     otherwise in EMPIDS_OVER;
~As: starting map restructure...
~As: REORGANIZE needed to preserve strict partitioning
~As: NO REORGANIZE was used to override scan
~As: reads: async 0 synch 21, writes: async 7 synch 3
SQL>
SQL> show storage map EMPLOYEES_MAP
EMPLOYEES_MAP
For Table:      EMPLOYEES
Placement Via Index:  EMPLOYEES_HASH
Partitioning is:  NOT UPDATABLE
Strict partitioning was not validated for this table
Comment:        employees partitioned by "00200" "00400"
Store clause:   STORE
                using (EMPLOYEE_ID)
                in EMPIDS_LOW
                with limit of ('00200')
                in EMPIDS_MID
                with limit of ('00400')
                otherwise in EMPIDS_OVER
Compression is:  ENABLED
SQL>

```

Entering a subsequent ALTER STORAGE MAP...REORGANIZE will validate the partitioning:

```

SQL> alter storage map EMPLOYEES_MAP
cont>     partitioning is not updatable
cont>     reorganize;
~As: starting map restructure...
~As: starting REORGANIZE...
~As: reorganize AREAS...
~As: processing rows from area 69
~As: processing rows from area 70
~As: processing rows from area 71
~As: reads: async 408 synch 22, writes: async 3 synch 0
SQL>

```

Usage Notes

- The NO REORGANIZE clause is ignored unless used with PARTITIONING IS NOT UPDATABLE. This is because either no automatic reorganize is required, or a full rebuild of the table is needed to implement the new map structure.
- REORGANIZE and NO REORGANIZE may not appear in the same ALTER STORAGE MAP command.


```

SQL> alter storage map EMPLOYEES_MAP
cont>   partitioning is not updatable
cont>   no reorganize
cont>   reorganize areas
cont>   store
cont>   using (EMPLOYEE_ID)
cont>   in EMPIDS_LOW
cont>     with limit of ('00200')
cont>   in EMPIDS_MID
cont>     with limit of ('00400')
cont>   otherwise in EMPIDS_OVER;
%SQL-F-MULTSPECATR, Multiple specified attribute. "REORGANIZE" was specified
more than once

```

- The SET FLAGS option STOMAP_STATS will output an indication that NO REORGANIZE was used.
- The SHOW STORAGE MAP command will output an indication that NO REORGANIZE was used. For example:

```

SQL> show storage map EMPLOYEES_MAP
EMPLOYEES_MAP
For Table:           EMPLOYEES
Placement Via Index: EMPLOYEES_HASH
Partitioning is:    NOT UPDATABLE
Strict partitioning was not validated for this table ...

```

7.4.4 New Options for the GET DIAGNOSTICS Statement

For Oracle Rdb7 Release 7.0.3.1, two new options have been added to the GET DIAGNOSTICS statement:

- TRANSACTION_TIMESTAMP

This option returns the date and time that the last transaction was started. If a transaction is not active, then this may be a prior transaction. The database server will start transactions when performing database operations and, therefore, this timestamp may reflect the time of an internal transaction.

If the default date format is SQL92, this option returns a value with the data type `TIMESTAMP(2)`; otherwise, it returns a `DATE (VMS)` data type. The default date format can be changed using either the `SET DIALECT` or `SET DEFAULT DATE FORMAT` statements, or one of the associated module attributes.

- TRANSACTION_SEQUENCE

This is the transaction sequence number (TSN) assigned to the most recently started transaction. The TSN is a unique indicator of database transaction activity; however, note that the TSN may be reused in some cases. The TSN for a read-only transaction reflects the transaction state which is visible to the transaction, and therefore it could have been previously assigned to a read/write transaction. If a read/write transaction performs no database I/O, or was rolled back, then that TSN may be reused by a subsequent read/write transaction.

This option returns a `BIGINT` data type.

The following example uses both these new options:

```
SQL> set transaction read write;
SQL> show transaction
Transaction information:
    Statement constraint evaluation is off
On the default alias
Transaction characteristics:
    Read Write
Transaction information returned by base system:
a read-write transaction is in progress
- updates have not been performed
- transaction sequence number (TSN) is 0:256
- snapshot space for TSNs less than 0:256 can be reclaimed
- recovery unit journal filename is USER2:[RDM$RUJ]SCRATCH$00018679B3AD.RUJ;1
- session ID number is 8
SQL>
SQL> declare :x date vms;
SQL>
SQL> begin get diagnostics :x = transaction_timestamp; end;
SQL> print :x;
X
27-MAY-1999 22:39:17.02
SQL>
SQL> declare :y bigint;
SQL>
SQL> begin get diagnostics :y = transaction_sequence; end;
SQL> print :y;
Y
256
SQL>
SQL> select current_timestamp from rdb$database;
27-MAY-1999 22:39:18.20
1 row selected
SQL>
SQL> commit;
```

7.4.5 RMU/SHOW Statistic OPCOM Message Tracking

The RMU/SHOW Statistic utility has been enhanced to provide tracking of database-related OPCOM messages. OPCOM messages are tracked using two independent methods:

1. New “OPCOM Messages” Screen. Located in the “Process Information” submenu, the “OPCOM Messages” screen identifies the last broadcast OPCOM message for each active process. If a single process is attached to the database multiple times, the OPCOM messages are displayed for the attach that issued the broadcast.
2. New /OPCOM_LOG qualifier. This qualifier specifies the file specification of the log file to record various OPCOM messages broadcast by attached database processes.

The following is an example of the “OPCOM Messages” screen:

```

Oracle Rdb X7.0-00 Performance Monitor OPCOM Log
Database KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS.RDB;2
OPCOM Log created 7-JUN-1999 06:25:10.61
06:25:12.13 2A53D804:1 Opening "KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS2.AIJ;1"
(missed 4)
06:25:12.13 2A526A05:1 AIJ Log Catch-Up Server activated (missed 4)
06:25:21.07 2A53D804:1 Opening "KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS3.AIJ;1"
06:27:50.11 2A53D804:1 After-image journal 14 switch-over in progress (to 15)
06:27:51.26 2A53D804:1 After-image journal switch-over complete
06:28:53.44 2A47061D:1 AIJ backup operation started
06:28:59.00 2A53D804:1 Automatic backup utility cannot be invoked for TCS3
ABS AIJ backup
06:30:29.27 2A53D804:1 After-image journal 15 switch-over in progress (to 16)
06:30:30.41 2A53D804:1 After-image journal switch-over complete (missed 2)
06:31:40.23 2A53DE21:1 AIJ backup operation started
06:31:54.56 2A53D804:1 Automatic backup utility cannot be invoked for TCS4
ABS AIJ backup
06:32:20.88 2A53DE21:1 AIJ backup operation completed
06:34:17.52 2A53D804:1 Opening "KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS1.AIJ;1"
06:36:56.55 2A51E222:1 AIJ backup operation started
06:37:30.46 2A51E222:1 AIJ backup operation completed
06:37:44.74 2A53D804:1 Replication server stalled waiting for MSN 5700 (AIJ
17:12526)
06:37:44.74 2A526A05:1 Replication server stalled waiting for MSN 5700 (AIJ
17:12526)
06:41:38.39 2A53D34B:1 After-image journal 17 switch-over in progress (to 18)
06:42:13.87 2A53D804:1 Opening "KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS2.AIJ;1"
06:42:15.49 2A53D34B:1 After-image journal switch-over complete (missed 2)
06:42:19.96 2A53D34B:1 After-image journal 18 switch-over in progress (to 19)
06:42:23.31 2A53D804:1 Opening "KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS3.AIJ;1"
06:42:23.31 2A53D34B:1 After-image journal switch-over complete
06:42:46.27 2A4D3423:1 AIJ backup operation started
06:44:26.61 2A53D804:1 Automatic backup utility cannot be invoked for TCS2
ABS AIJ backup
06:44:32.13 2A53D804:1 Automatic backup utility cannot be invoked for TCS2
ABS AIJ backup
06:44:33.27 2A53D804:1 Opening "KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS4.AIJ;1"
(missed 2)
06:44:34.45 2A53D804:1 After-image journal switch-over complete
06:44:34.45 2A532216:31 Automatic backup utility cannot be invoked for TCS2
ABS AIJ backup
06:44:35.66 2A4F9A1A:1 Automatic backup utility cannot be invoked for TCS2
ABS AIJ backup
06:44:35.66 2A461220:1 Automatic backup utility cannot be invoked for TCS2
ABS AIJ backup
06:44:36.88 2A505617:1 Automatic backup utility cannot be invoked for TCS2
ABS AIJ backup
06:44:36.88 2A517218:1 Automatic backup utility cannot be invoked for TCS2
ABS AIJ backup
06:44:36.88 2A462C1F:1 Automatic backup utility cannot be invoked for TCS2
ABS AIJ backup
06:44:40.29 2A46EC1C:1 Automatic backup utility cannot be invoked for TCS2
ABS AIJ backup
06:44:41.45 2A4EBA19:1 Automatic backup utility cannot be invoked for TCS2
ABS AIJ backup
06:44:43.71 2A4F361E:1 Automatic backup utility cannot be invoked for TCS2
ABS AIJ backup
06:46:10.60 2A53E224:1 AIJ backup operation started
06:46:20.53 2A53D804:1 Automatic backup utility cannot be invoked for TCS3
ABS AIJ backup
06:46:42.36 2A53E224:1 AIJ backup operation completed
06:47:11.60 2A53D804:1 Opening "KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS5.AIJ;1"
06:48:16.92 2A4D7825:1 AIJ backup operation started
06:48:47.36 2A4D7825:1 AIJ backup operation completed
06:49:40.60 2A53D804:1 Opening "KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS1.AIJ;1"

```

```

(missed 2)
06:49:41.76 2A53D804:1 After-image journal switch-over complete
06:52:45.64 2A53D34B:1 After-image journal 22 switch-over in progress (to 23)
06:52:50.04 2A53D804:1 Opening "KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS2.AIJ;1"
06:52:50.04 2A53D34B:1 After-image journal switch-over complete
06:53:49.82 2A474E26:1 AIJ backup operation started
06:54:29.37 2A53D804:1 Automatic backup utility cannot be invoked for TCS1
ABS AIJ backup
06:55:14.93 2A53D804:1 Automatic backup utility cannot be invoked for TCS1
ABS AIJ backup
06:55:29.91 2A53D804:1 Automatic backup utility cannot be invoked for TCS1
ABS AIJ backup

```

The following is an example of the /OPCOM_LOG qualifier log file contents:

```

Oracle Rdb X7.0-00 Performance Monitor OPCOM Log
Database KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS.RDB;2
OPCOM Log created 4-JUN-1999 14:52:16.81
14:52:26.87 2A506125:1 After-image journal 33 switch-over in progress (to 34)
14:52:28.44 2A506125:1 After-image journal switch-over complete
14:53:40.79 2A506125:1 After-image journal 34 switch-over in progress (to 35)
14:53:42.22 2A506125:1 After-image journal switch-over complete
15:03:21.67 2A506125:1 After-image journal 36 switch-over in progress (to 37)
15:03:23.16 2A506125:1 After-image journal switch-over complete
15:03:24.82 2A459220:1 AIJ backup operation started
15:03:27.82 2A459220:1 AIJ backup operation completed

```

When recording OPCOM messages, it is possible to occasionally miss a few messages for a specific process. When this occurs, the message “n missed” will be displayed in the log file.

It is possible to record specific operator classes of OPCOM messages if you specify the /OPTION=VERBOSE qualifier. This qualifier records only those messages that can be received by the process executing the RMU/SHOW Statistic utility. For example, if the process is enabled to receive operator class CENTRAL, then specifying /OPCOM_LOG=opcom.log/OPTION=VERBOSE will record all CENTRAL operator messages. Conversely, specifying only the /OPCOM_LOG=opcom.log qualifier will record all database-specific OPCOM messages generated from this node.

The operator-specific log file output format is different from the database-specific contents, because the output is being captured directly from OpenVMS. The following is an example of the operator-specific log file contents for the CLUSTER and CENTRAL operator classes:

```

Oracle Rdb X7.0-00 Performance Monitor OPCOM Log
Database KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS.RDB;2
OPCOM Log created 11-JUN-1999 10:52:07.53
11-JUN-1999 10:52:23.85) Message from user RDBVMS on ALPHA4 Oracle Rdb
X8.0-00 Event Notification for Database _$111$DUA368:[BBENTON.TEST]
MF_PERSONNEL.RDB;1 AIJ Log Server terminated
11-JUN-1999 10:52:25.49) Message from user RDBVMS on ALPHA4 Oracle Rdb
X8.0-00 Event Notification for Database _$111$DUA368:[BBENTON.TEST]
MF_PERSONNEL.RDB;1 AIJ Log Roll-Forward Server started
11-JUN-1999 10:52:26.06) Message from user RDBVMS on ALPHA4 Oracle Rdb
X8.0-00 Event Notification for Database _$111$DUA368:[BBENTON.TEST]
MF_PERSONNEL.RDB;1 AIJ Log Roll-Forward Server failed
.
.
11-JUN-1999 10:54:16.05) Message from user INTERNet on ALPHA4 TELNET Login
Request from Remote Host: 138.2.136.180 Port: 1252
11-JUN-1999 10:54:16.36) Message from user RDBVMS on ALPHA4 Oracle Rdb
X8.0-00 Event Notification for Database _$111$DUA368:[BBENTON.TEST]
MF_PERSONNEL.RDB;1 AIJ Log Catch-Up Server terminated

```

7.4.6 New Restricted_Access Qualifier for RMU/LOAD

RMU Load now supports the `Restricted_Access` qualifier when attaching to an Oracle Rdb database. This option allows a single process to load data and enables some optimizations available only when `Restricted Access` is in use.

If you are loading a table from an RMU Unload file which contains `LIST OF BYTE VARYING` data then the `Restricted_Access` qualifier will reserve the `LIST` areas for exclusive access. This reduces the virtual memory used by long transactions in RMU Load and also eliminates I/O to the snapshot files for the `LIST` storage areas.

The `Restricted_Access` and `Parallel` qualifiers are mutually exclusive and may not both be specified on the RMU Load command line, or within a plan file. While RMU Load is running with this option enabled, no other user may attach to the database. The default is `Norestricted_Access`.

7.4.7 RDO EDT Editor on OpenVMS Alpha Now Available

Previously, on OpenVMS Alpha, the RDO editor was restricted to the TPU editor. The EDT editor was not available via the `RDO$EDIT` logical name.

This problem has been corrected. The `RDO$EDIT` logical name controls the editor selection between EDT and TPU as it does on OpenVMS VAX.

7.4.8 New Options Added to SQL EXTRACT Function

In Oracle Rdb7 Release 7.0.3.1, the SQL `EXTRACT` function is being enhanced with two new options: `WEEK_NUMBER` and `YEAR_WEEK`. These options return the week number as defined by the International Standard ISO 8601:1988 "Data elements and interchange formats - Information interchange - Representation of dates and times".

Section 3.17 of this standard defines a week as "week, calendar: A seven day period within a calendar year, starting on a Monday and identified by its ordinal number within a year; the first calendar week of the year is the one that includes the first Thursday of that year. In the Gregorian calendar, this is equivalent to the week which includes 4 January."

`WEEK_NUMBER` is a number between 1 and 53 representing the week of the year (most years only have 52 weeks). A week starts on Monday and has most of its days falling in a specific year.

YEAR_WEEK is a variation of the WEEK_NUMBER that includes the year (including the century) in which the week logically falls. The values range from 185901 through 999952 (higher values are possible if dates are constructed with a year beyond 9999). The last two digits of the value are identical to the value returned by the WEEK_NUMBER option.

The following example shows the new function results:

```
SQL> select dt,
cont>      extract (week_number from dt),
cont>      extract (year_week from dt)
cont> from week_sample
cont> order by dt;
DT
1859-01-07          1          185901
1999-01-01          53          199853
1999-01-04          1          199901
1999-01-10          1          199901
1999-12-31          52          199952
2000-01-01          52          199952
2000-01-03          1          200001
2000-02-28          9          200009
2000-02-29          9          200009
2000-03-01          9          200009
9999-12-31          52          999952
11 rows selected
```

Usage Notes

- The source date/time expression must include a date component; DATE (ANSI), TIMESTAMP, or DATE (VMS). Attempts to use other data types will result in an error. For example:

```
SQL> select extract (week_number from current_time) from ...;
%RDB-E-CONVERT_ERROR, invalid or unsupported data conversion
-RDMS-E-EXT_WEEKDAY_TS, invalid type for EXTRACT - must be DATE or TIMESTAMP
```

- Note that neither WEEK_NUMBER nor YEAR_WEEK can be calculated for the year 1858, or the first few days of 1859 because the Oracle Rdb date only supports part of the year 1858, and therefore the calculation cannot be made. For example:

```
SQL> select extract (week_number from date'1859-1-1') from ...;
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-COSI-F-IVTIME, invalid date or time
```

- Note that the year in which the week falls may not be the same as the extracted year from the date/time value because days at the start of a calendar year may logically fall in the last week of the previous year, and days at the end of a calendar year may logically fall in the first week of the following year.

7.5 Enhancements Provided in Oracle Rdb7 Release 7.0.2.1

7.5.1 New RMU Extract ITEM=REVOKE_ENTRY

With Oracle Rdb7 Release 7.0.2, RMU Extract now supports an option for the ITEM qualifier. This item, REVOKE_ENTRY, allows the database administrator to extract a SQL (or RDO) script which deletes the protections from all access control lists in the database (database, table, column, module, function and procedure).

The output contains a series of SQL REVOKE ENTRY statements (or DELETE PROTECTION statements if the language selected is RDO) which remove the access control entry for the user and all objects.

The following sample shows the output from the new REVOKE_ENTRY item.

```
$ RMU/EXTRACT/ITEM=REVOKE_ENTRY ACCOUNTING_DB
.
.
.
--                               Protection Deletions
--
-----
revoke entry
  on database alias RDB$DBHANDLE
  from [RDB,JAIN];
revoke entry
  on database alias RDB$DBHANDLE
  from [RDB,SMITHI];
revoke entry
  on database alias RDB$DBHANDLE
  from PUBLIC;
revoke entry
  on table ACCOUNT
  from [RDB,SMITHI];
revoke entry
  on table ACCOUNT
  from PUBLIC;
revoke entry
  on table ACCOUNT_BATCH_PROCESSING
  from [RDB,SMITHI];
revoke entry
  on table ACCOUNT_BATCH_PROCESSING
  from PUBLIC;
revoke entry
  on table BILL
  from [RDB,SMITHI];
revoke entry
  on table BILL
  from PUBLIC;
.
.
.
```

This note was inadvertently left out of the New Features section of the 7.0.2 Release Notes.

7.5.2 New Data Type Support for DEC FORTRAN

In previous versions of the SQL precompiler, the use of INTEGER*1, INTEGER*8 or LOGICAL*8 types would result in an error as shown below.

```
$ sql$pre /for INTEGER8.SFO
      2          into :salary indicator :ind_var
      1
%SQL-F-INVHVDECL, (1) Host variable SALARY was invalidly declared.
```

For all platforms, the SQL Precompiler now supports INTEGER*1 (8 bit binary) which is identical to the BYTE and LOGICAL*1 FORTRAN types which are currently supported. This FORTRAN type is similar to the unscaled SQL TINYINT data type.

For OpenVMS Alpha, the SQL Precompiler now supports INTEGER*8 and LOGICAL*8 (64 bit binary). These FORTRAN types are similar to the unscaled SQL BIGINT data type.

Table 7–1 Supported FORTRAN Datatypes

FORTRAN type	SQL type	Comments and Restrictions
BYTE	TINYINT	
CHARACTER*n	CHAR	The n represents a positive integer literal
INTEGER	INTEGER	
INTEGER*1	TINYINT	
INTEGER*2	SMALLINT	
INTEGER*4	INTEGER	
INTEGER*8	BIGINT	OpenVMS Alpha only
LOGICAL	INTEGER	
LOGICAL*1	TINYINT	
LOGICAL*2	SMALLINT	
LOGICAL*4	INTEGER	
LOGICAL*8	BIGINT	OpenVMS Alpha only
REAL	REAL	
REAL*4	REAL	
REAL*8	DOUBLE PRECISION	
STRUCTURE /name/ integer*2 len character*n body END STRUCTURE	VARCHAR	The named structure can be used to define other FORTRAN host variables. The len component of the structure must be set to the correct length of the string before use as a parameter to SQL. The n represents a positive integer literal.

This note was inadvertently left out of the New Features section of the 7.0.2 Release Notes.

7.5.3 New Data Type Support for DEC C

Bug 427695

In previous versions of the SQL precompiler, the use of the data type `__int64`, `__int32` and `__int16` types would result in an error as shown below.

```
$ sql$pre int16.sc/cc
      into :salary indicator :ind_var
      1
%SQL-F-HVNOTDECL, (1) Host variable ind_var was not declared
```

These data types are now supported for DEC C in Oracle Rdb7 Release 7.0.2.

In addition, several type definitions (typedef) are provided on OpenVMS Alpha when using the `ints.h` header file. These type definitions are now also supported by the SQL precompiler: `int8`, `int16`, `int32` and `int64`.

For both OpenVMS VAX and Alpha platforms, the SQL precompiler now supports the types: `int8`, `int16`, `__int16`, `int32`, and `__int32`.

For OpenVMS Alpha, the SQL Precompiler also supports int64 and __int64 (64 bit binary).

Table 7-2 Supported C Datatypes

C type or typedef	SQL type	Comments and Restrictions
char	CHARACTER	
int	INTEGER	
short	SMALLINT	
float	REAL	
double	DOUBLE PRECISION	
enum	INTEGER	
long	INTEGER or BIGINT	On OpenVMS the data type long is 32 bits, and on Digital UNIX data type long is 64 bits
int8	TINYINT	Requires #include <ints.h>
int16	SMALLINT	Requires #include <ints.h>
__int16	SMALLINT	
int32	INTEGER	Requires #include <ints.h>
__int32	INTEGER	
int64	BIGINT	OpenVMS Alpha platform only. Requires #include <ints.h>
__int64	BIGINT	OpenVMS Alpha platform only

This note was inadvertently left out of the New Features section of the 7.0.2 Release Notes.

7.5.4 RMU/SHOW STATISTIC "TSNBLK Allocation" Screen

The RMU/SHOW STATISTIC utility has been enhanced to provide runtime information about the allocation of TSNBLK entries in the database rootfile.

Each TSNBLK entry controls the allocation of processes to nodes, and records the allocation and state of each process' TSN information. Consequently, each TSNBLK entry contains a lot of valuable information about the runtime state of the database.

The "TSNBLK Allocation" screen is located in the "Database Parameter Info" sub-menu. The screen is not recorded in the binary output file, nor is it available when replaying a binary input file.

The "TSNBLK Allocation" screen displays cluster-wide information. However, information concerning nodes other than the current node may occasionally become stale, as the RMU/SHOW STATISTIC utility does not automatically refresh the cluster information; it relies on application processes to do this. If you believe the information displayed is stale, use the "Refresh" on-screen menu option to force a refresh of the cluster information; however, this is seldom necessary.

The following is an example of the "TSNBLK Allocation" screen.

```
Node: ALPHA3 (1/4/24)   Oracle Rdb X7.0-xx Perf. Monitor 27-APR-1999 08:11:25.32
Rate: 1.00 Second      TSNBLK Allocation                               Elapsed: 00:14:12.85
Page: 1 of 2           KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS.RDB;2       Mode: Online
```

```
-----
```

TSNBLK	Nodename.	#RwTx	#RoTx	#NoTx	#Srvr	#Busy	#Free	CommitTSN	OldestTSN
0	BONZAI	0	0	0	2	2	26	0:2651	0:0
1	ALPHA3	9	0	0	1	10	18	0:3203	0:2564
2	ALPHA5	10	0	0	1	11	17	0:3202	0:2531
3	ALPHA4	9	0	0	1	10	18	0:3195	0:2386
4	available								
5	available								
6	available								
7	available								
8	available								
9	available								
10	available								
11	available								
12	available								
13	available								
14	available								
15	available								
16	available								

```
-----
```

```
Exit Help Menu >next_page <prev_page Options Refresh Set_rate Write Zoom !
```

The "TSNBLK Allocation" screen displays one line of information for each TSNBLK entry. Each TSNBLK entry contains the following fields:

- **TSNBLK.** This field identifies the TSNBLK entry number for easy identification. This is *not* the block number where the TSNBLK entry resides in the database rootfile.
- **Nodename.** This field identifies the name of the node to which a TSNBLK is allocated. If the TSNBLK is not allocated to any node, the name "available" is displayed.

Once a TSNBLK entry has been allocated to a particular node, it remains allocated to that node until all processes assigned to the TSNBLK have detached from the database.

- **#RwTx.** This field identifies the total number of processes assigned to the TSNBLK entry that have an active read-write transaction.
- **#RoTx.** This field identifies the total number of processes assigned to the TSNBLK entry that have an active read-only transaction.
- **#NoTx.** This field identifies the total number of idle processes assigned to the TSNBLK entry. An idle process is one which does not have an active transaction.

This field ideally should always display the value "0".

- **#Srvr.** This field identifies the total number of database server processes assigned to the TSNBLK entry. Database server processes include the ABS, ALS, LCS, LRS, RCS and DBR processes.
- **#Busy.** This field identifies the total number of active processes assigned to the TSNBLK entry. The value of this field includes server processes.
- **#Free.** This field identifies the total number of processes that remain to be assigned to the TSNBLK entry.

- **CommitTSN.** This field identifies the last commit transaction TSN for this TSNBLK entry.
- **OldestTSN.** This field identifies the oldest active transaction TSN for this TSNBLK entry.

In general, the following calculations are interesting:

- $\text{possible_processes} = \text{\#Busy} + \text{\#Free}$
- $\text{user_processes} = \text{\#Busy} - \text{\#Srvr}$

To display additional information about the processes assigned to a particular TSNBLK entry, use the "Zoom" on-screen menu option to select the desired TSNBLK entry. The following is an example of the information displayed by the TSNBLK zoom screen:

```
-----
Active user with process ID 2A421B54 (database server)
  User name is RDBVMS
  Process name is RDM_ALS70_1
  Image name is DSA1:[SYS4.SYSCOMMON.][SYSEXE]RDMALS70.EX
  No transaction in progress
  Monitor ID is 2
  Internal Stream ID is 1
  Internal Transaction ID is 1310
Active user with process ID 2A430356
  User name is R_ANDERSON
  Process name is RICK5
  Image name is $111$DUA616:[RDMS_X07100.VAX.][RMU]RMU70.
  Read/write transaction in progress
  Transaction sequence number is 0:6192
-----
```

7.5.5 Scheduled AIJ Switch-overs

Often it is desirable to perform an AIJ switch-over operation prior to the start of a known "busy" period, such as end-of-day processing. Having a new AIJ journal available during the busy period minimizes the risk of an unplanned AIJ switch-over occurring during the busy period, possibly adversely affecting system performance during critical operations.

Currently, automatic AIJ switch-over operations can be performed by the DBA using operating system tools executing the RMU/SET AFTER/SWITCH_JOURNAL command. However, this is a workaround solution at best.

Oracle Rdb7 now provides a method to schedule automated AIJ switch-over operations when using the ALS process. The logical name RDM\$BIND_ALS_SCHEDULE_SWITCH, located in the LNM\$SYSTEM_TABLE name table, can be used to define up to 96 different times, using 15 minute intervals, for an automatic AIJ switch-over to occur.

When using multiple databases on a given node, Oracle recommends you deassign the logical after each database has been opened. Otherwise, all databases on the node will perform AIJ switch-over operations at the same time, which may have undesirable system performance characteristics.

The RDM\$BIND_ALS_SCHEDULE_SWITCH logical name value consists of a comma-separated list of times, hour and/or minutes, when the AIJ switch-over should be performed. The logical name value format is the following:

```
( [ HOUR ] [ : [ MINUTE ] ] [ , ... ] )
```

Hours must be in the range from “0” to “23”; for example, an hour value of “1” is 1AM while a value of “13” is 1PM.

Minutes must be in the range from “0” to “59”, and are rounded *down* to the nearest quarter-hour; for example, a minute value of “50” is treated as “45”.

The ALS process evaluates the scheduling values when it is started up. If you change the logical value, you must stop and restart the ALS process. Note that this is not possible if you are using the Hot Standby feature. The ALS process displays the scheduled AIJ switch-over times in the output file.

Be careful when using the RDM\$BIND_ALS_SCHEDULE_SWITCH logical name on multiple nodes. If you schedule an AIJ switch-over operation to occur at the same time on multiple nodes, then multiple AIJ switch-over operations will occur simultaneously, possibly resulting in an empty AIJ journal.

The RMU/SHOW STATISTIC “AIJ Journal Information” screen displays the total number of scheduled AIJ switch-overs. The actual switch-over times are not displayed. For example, the **SwchSched** field indicates that there are 13 AIJ switch-over operations currently scheduled.

```
Node: MYNODE (1/1/24) Oracle Rdb X7.0-00 Perf. Monitor 20-MAY-1999 09:16:10.61
Rate: 1.00 Second AIJ Journal Information Elapsed: 16:02:57.94
Page: 1 of 1 MY_DISK:[R_ANDERSON.TCS_MASTER]TCS.RDB;2 Mode: Online
```

```
-----
Journaling: enabled Shutdown: 120 Notify: enabled State: Accessible
ALS: Running ABS: enabled ACE: disabled FC: enabled CTJ: disabled
ARB.Count: 300 ARB.Avail: 300 SwchSched: 13
After-Image.Journal.Name..... SeqNum AIJsize CurrEOF Status. State.....
TCS1 22 100002 2 Current Accessible
TCS2 Unused 100100 Empty Latent Accessible
TCS3 Unused 10200 Empty Latent Accessible
TCS4 Unused 10300 Empty Latent Accessible
TCS5 Unused 10400 Empty Latent Accessible
Available AIJ slot 1
Available AIJ slot 2
-----
```

```
Bell Exit Help Menu >next_page <prev_page Refresh Set_rate Write Zoom !
```

The following example shows how to schedule an AIJ switch-over to occur automatically every half-over from 8AM to 3:30PM.

```
$ DEFINE/SYS RDM$BIND_ALS_SCHEDULED_SWITCH -
"8,9,10,11,12,13,14,15,8:30,9:30,10:30,11:30,12:30,13:30,14:30,15:30"
```

The following table describes some different formats and their meanings:

RDM\$BIND_ALS_SCHEDULED_SWITCH	Meaning
8	8AM
9:	9AM
:30	12:30AM
15:00	3PM
17:45	5:45PM
:	12 Midnight

NOTE: It is recommended that you do **not** use this logical name on any database that serves as a standby of some other master database. If HS is shutdown and an AIJ switch-over occurs on the standby database, HS will be unable to restart replication because the standby database will be considered "modified".

7.5.6 New Logicals for RMU/OPTIMIZE/AFTER_JOURNAL

RDM\$BIND_OPT_TXN_THRESHOLD may be defined to limit the number of times a transaction with a record larger than the sort record size or a DDL record will log an informational message and cause multiple small sorts, reducing performance and efficiency of optimizing the AIJ journal. The default is 10 per transaction. The minimum is 0, maximum is "unlimited", and default is 10.

RDM\$BIND_OPT_SORT_THRESHOLD may be defined to limit the number of records sorted at one time, which reduces the size of the sort workfiles when disk space is limited. The efficiency of optimizing the journal is not as good as sorting all records in one pass, but the sort may be faster and disk space for sort workfiles is greatly reduced. The optimized file will be a little larger if the sort is limited to a specified number of records. The minimum is 0, and the maximum and default are both "unlimited." Specifying zero means "unlimited".

These new logicals are available in Oracle Rdb7 Release 7.0.2.1.

7.6 Enhancements Provided in Oracle Rdb7 Release 7.0.2

7.6.1 Query Performance and Cartesian Limit

Bugs 551177, 604552 and 620579

The elapsed time it takes to perform a query is made up of two parts. The first part is the time it takes the Rdb optimizer to sift through many possible strategies in order to come up with one which gives the best performance. The second part is the time it takes to execute the query under the chosen strategy.

Often, little consideration is given to the optimizer execution time, either because that time is negligibly small or because it is small compared to the total query execution time. Also, after a query has been compiled, it can be executed many times. In such cases, it might be worthwhile to pay the cost of finding the best query strategy since that cost is incurred only the first time the query is executed. One can avoid the optimization cost by using a query outline but use of and maintenance of query outlines is not always a practical alternative.

In searching for a query solution that gives the best query performance, the Rdb optimizer tries many permutations when joining multiple tables. For performance reasons, it is desirable to ignore solutions that generate large cartesian products. To achieve this the Rdb optimizer, prior to Rdb 7.0, always extended a partial join order with a join item (a table or a sub-query) that shared a join column with an already placed join item. If a table were small, however, this could lead to sub-optimal join orders, as in the case of 'star join' queries. In Rdb 7.0, the optimizer was changed to allow small tables to be placed anywhere in the join order, provided that a query outline is not present.

Allowing small tables to be placed anywhere in a join order tends to increase the number of solutions tried by the optimizer. This in turn increases the time it takes for the optimizer to decide on a final solution. Therefore we have two competing effects. On the one hand, query execution time might be improved because a better solution is found. On the other hand, the first time the query is performed, execution might be slower due to the time it takes to examine all possible solutions.

Now there is a way for the user to reach a balance between these two effects. The user can instruct Rdb to limit the number of small tables that it will allow to be placed anywhere within a join order. This can be done by using the SQL

statement `SET FLAGS`, or on OpenVMS systems by defining the logical name `RDMS$SET_FLAGS`.

Format for the SQL `SET FLAGS` statement:

```
SET FLAGS 'CARTESIAN_LIMIT(n)';
```

Format for the OpenVMS `DEFINE` command:

```
$ DEFINE RDMS$SET_FLAGS "CARTESIAN_LIMIT(n)"
```

The keyword `CARTESIAN_LIMIT` can be abbreviated as `CART`. If the value `n` is omitted, the default value is five. If the `CARTESIAN_LIMIT` flag is not set, the default limit is five. The limit can be anywhere from 0 to 128, the maximum number of tables that Rdb allows to be joined per query. Actually, the cartesian limit can be higher than 128; but that has no practical value since it will be treated as being equivalent to 128. If one uses the negated form of the keyword, e.g., `SET FLAGS 'NOCART'`, the limit is set to zero.

A limit of zero does not mean that cartesian products are disallowed entirely. Versions of Rdb earlier than 7.0 allowed cartesian products, only in a more restricted sense than does Rdb 7.0. If you specify a limit of zero, the optimizer behavior will be approximately the same as prior to Rdb 7.0. That is, restrictions on the placement of a table within a join order will apply to all tables whether large or small.

A limit of 128 (or higher) means that the behavior will be that of Rdb 7.0. All small tables will be placeable anywhere within a join order, provided that a query outline is not used.

If you find that a query takes longer to execute than you'd like, you might be able to reduce execution time by experimenting with the `SET FLAGS 'CARTESIAN_LIMIT(n)'` statement. Lower the value of `n` below five if the query performs slowly only the first time it is executed. If your query has characteristics similar to those mentioned earlier and if the query consistently performs slowly (not just the first time it is compiled), try raising the limit above five to see if a better query strategy is chosen.

7.6.2 Named Query Outlines May Now Be Used by Triggers and Constraints

This enhancement was included in Oracle Rdb 6.1 but the release note was inadvertently left out.

During compilation of a constraint or trigger, Oracle Rdb will now search for a query outline with the same name as the trigger or constraint being compiled. If a match is found, that outline will be used during the query compilation of the trigger or constraint. If no outline is found matching the name of the object, Oracle Rdb will then try to locate an appropriate outline using the BLR ID of the query.

For example:

```

.
.
.
SQL> create table TAB1 (a1 int constraint TAB1NOTNULL not null ,
cont> cont>                a2 char(10),
cont> cont>                a3 char(10) );
SQL> create outline TAB1NOTNULL
cont> id '8755644ECB040948E28A76B6D77CC2D3'
cont> mode 0
cont> as (
cont>   query (
cont>     subquery (
cont>       TAB1 0  access path sequential
cont>     )
cont>   )
cont> )
cont> compliance optional      ;
SQL> create trigger TAB1TRIG before insert on TAB1
cont> (update TAB1 set a3= 'bbbb' where a2 = 'aaaa' ) for each row;
SQL> create outline TAB1TRIG
cont> id '990F90B45658D27D64233D88D16AD273'
cont> mode 0
cont> as (
cont>   query (
cont>     subquery (
cont>       TAB1 0  access path sequential
cont>     )
cont>   )
cont> )
cont> compliance optional      ;
.
.
.
$ DEFINE RDMS$DEBUG_FLAGS "SnsI"
.
.
.
SQL> insert into tabl (a1) value (11);
~S: Trigger name TAB1TRIG
~S: Outline TAB1TRIG used
~S: Outline TAB1NOTNULL used
Conjunct      Get      Retrieval sequentially of relation TAB1
.
.
.
1 row inserted
SQL> commit;
~S: Constraint TAB1NOTNULL evaluated
Conjunct      Get      Retrieval sequentially of relation TAB1
.
.
.

```

7.6.3 GRANT and REVOKE Support * for Object Names

With the release of Oracle Rdb7 Release 7.0.2, both the GRANT and REVOKE statement now accept an asterisk in place of the list of database alias, table, module, function, or procedure names. This asterisk selects all objects of the specified type.

```

SQL> ! allow all access to use JAIN
SQL> grant select on database alias * to jain;
SQL> grant select on table * to jain;
SQL> grant execute on module * to jain;
SQL> grant execute on procedure * to jain;
SQL> grant execute on function * to jain;

```

If privileges are denied for the operation on some objects then the GRANT or REVOKE action is aborted, however, some objects may have protection changes applied.

In prior releases, the GRANT and REVOKE statement specifying the ON TABLE * clause only operated on tables, and not also views as expected. This problem has been corrected in Oracle Rdb7 Release 7.0.2. With this release the * is applied to both table and view names.

7.6.4 New SET and SHOW DISPLAY Statements for Interactive SQL

Starting with Oracle Rdb7 Release 7.0.2, Interactive SQL supports the new SET DISPLAY statement to control the output of header information. A companion SHOW DISPLAY statement indicates the current settings.

```

--> SET DISPLAY +-----+-----+ EDIT STRING --+--+-->
                |         |         |         |         |
                | +-> NO  -+  +-> QUERY HEADER -+         |
                |         |         |         |         |
                |         |         |         |         |
                |         |         |         |         |
                |         |         |         |         |
                |         |         |         |         |
                |         |         |         |         |
                +-----+-----+ , <-----+-----+
--> SHOW DISPLAY -->

```

The SET DISPLAY command accepts options to enable or disable parts of the formatted output generated by various statements in Interactive SQL.

- EDIT STRING enables the usage of column edit strings to format values for SELECT. Use NO EDIT STRING to disable the use of the column's edit strings.
- QUERY HEADER enables the printed header generated by the SELECT, CALL, FETCH and PRINT statements. Use NO QUERY HEADER to disable this header.
- ROW COUNTER enables the total count reported by SELECT, DELETE, INSERT and UPDATE statements. Use NO ROW COUNTER to disable the trailing count message.

The defaults (as in previous versions) are to use edit strings, display the query header, and report a row count message. More than one option can be specified, separated by commas. However, you may not specify both the option and its negated form in the one command as demonstrated in the following example.

```

SQL> set display query header, no query header
%SQL-F-MULTSPECATR, Multiple specified attribute.
"QUERY HEADER" was specified more than once

```

The following example shows the effect of the SET DISPLAY statement and it uses the SHOW DISPLAY command to report the current settings.


```

SQL> attach 'file MF_PERSONNEL';
SQL>
SQL> create domain MONEY integer(2) edit string '$$$,$$9.99';
SQL> create table TEMP_EMP (id integer, sal MONEY);
SQL>
SQL> select * from work_status;
  STATUS_CODE  STATUS_NAME  STATUS_TYPE
  0            INACTIVE   RECORD EXPIRED
  1            ACTIVE     FULL TIME
  2            ACTIVE     PART TIME
3 rows selected
SQL>
SQL> set display no row counter;
SQL> show display
Output of the query header is enabled
Output of the row counter is disabled
Output using edit strings is enabled
HELP page length is set to 24 lines
Line length is set to 132 bytes
SQL>
SQL> select * from work_status;
  STATUS_CODE  STATUS_NAME  STATUS_TYPE
  0            INACTIVE   RECORD EXPIRED
  1            ACTIVE     FULL TIME
  2            ACTIVE     PART TIME
SQL> insert into TEMP_EMP (id) values (0);
SQL> insert into TEMP_EMP (id, sal)
cont>   select employee_id, max(salary_amount)
cont>   from salary_history group by employee_id;
SQL> update TEMP_EMP set id = NULL where id <= 0;
SQL> delete from TEMP_EMP where id is NULL;
SQL>
SQL> set display row counter;
SQL> show display
Output of the query header is enabled
Output of the row counter is enabled
Output using edit strings is enabled
HELP page length is set to 24 lines
Line length is set to 132 bytes
SQL>
SQL> select * from work_status;
  STATUS_CODE  STATUS_NAME  STATUS_TYPE
  0            INACTIVE   RECORD EXPIRED
  1            ACTIVE     FULL TIME
  2            ACTIVE     PART TIME
3 rows selected
SQL>
SQL> set display no query header;
SQL> show display
Output of the query header is disabled
Output of the row counter is enabled
Output using edit strings is enabled
HELP page length is set to 24 lines
Line length is set to 132 bytes
SQL>
SQL> declare :res integer;
SQL>
SQL> -- should omit query header for the SELECT statement
SQL> select * from work_status;
  0            INACTIVE   RECORD EXPIRED
  1            ACTIVE     FULL TIME
  2            ACTIVE     PART TIME
3 rows selected
SQL>
SQL> -- should omit query header for the PRINT statement

```

```

SQL> print :res;
      0
SQL> print 'This is a print line';
      This is a print line
SQL>
SQL> create module call_sample
cont>   language SQL
cont>   procedure ADD_ONE (in :a integer, out :b integer);
cont>     set :b = :a + 1;
cont> end module;
SQL> --should omit query header for the OUT/INOUT parameters for CALL
SQL> call ADD_ONE (100, :res);
      101

SQL>
SQL> declare c cursor for select * from work_status;
SQL> open c;
SQL> -- should omit query headers for the variables fetched
SQL> fetch c;
      0                INACTIVE          RECORD EXPIRED
SQL> set display query header;
SQL> show display
Output of the query header is enabled
Output of the row counter is enabled
Output using edit strings is enabled
HELP page length is set to 24 lines
Line length is set to 132 bytes
SQL> -- should output query headers for the variables fetched
SQL> fetch c;
      STATUS_CODE   STATUS_NAME   STATUS_TYPE
      1             ACTIVE          FULL TIME
SQL> close c;
SQL>
SQL> truncate table TEMP_EMP;
SQL> insert into TEMP_EMP (id, sal)
cont>   select employee_id, avg(salary_amount)
cont>   from salary_history
cont>   where salary_end is NULL
cont>   group by employee_id;
100 rows inserted
SQL>
SQL> select * from TEMP_EMP order by id limit to 3 rows;
      ID           SAL
      164      $51,712.00
      165      $11,676.00
      166      $18,497.00
3 rows selected
SQL>
SQL> set display no edit string;
SQL> show display
Output of the query header is enabled
Output of the row counter is enabled
Output using edit strings is disabled
HELP page length is set to 24 lines
Line length is set to 132 bytes
SQL>
SQL> select * from TEMP_EMP order by id limit to 3 rows;
      ID           SAL
      164      51712.00
      165      11676.00
      166      18497.00
3 rows selected
SQL>
SQL> set display edit string;
SQL> show display
Output of the query header is enabled

```

```

Output of the row counter is enabled
Output using edit strings is enabled
HELP page length is set to 24 lines
Line length is set to 132 bytes
SQL>
SQL> select * from TEMP_EMP order by id limit to 3 rows;
          ID          SAL
         164    $51,712.00
         165    $11,676.00
         166    $18,497.00
3 rows selected
SQL>
SQL> rollback;

```

Note

SHOW DISPLAY may also report the current line length which can be changed using the SET LINE LENGTH command, and the HELP page length which is automatically established for the interactive HELP command.

7.6.5 SQL Now Supports DBKEY String Literals

Oracle Rdb7 Release 7.0.2 adds a new format of literal for database keys. This feature is primarily of use to database administrators who have database keys which were displayed in error messages or shown on an RMU/SHOW STATISTICS display and wish to display the associated row.

The DBKEY string literal is prefixed by `_DBKEY`, or `_ROWID` to identify it as a special DBKEY literal. Some examples of valid DBKEY literals follow.

- `_DBKEY'23:5628:0'`

An Oracle Rdb table DBKEY has three parts, a logical area (in this example 23), a page number (in this example 5628), and a line number (in this example 0). All three parts must be specified.

- `_ROWID'23:5628:0, 45:345:15'`

The DBKEY string literal may include several comma separated DBKEYS if this is used to reference a view table. Each DBKEY references a row from the view made up of component rows from a table.

The ROWID keyword is a synonym for DBKEY.

Leading and trailing spaces are ignored, however spaces may not be embedded within the numeric values in the DBKEY.

Errors will be reported if the DBKEY is for a different table, is incorrectly formatted, or does not reference a row. The reported errors are shown in the following example. A question mark is placed within the string to highlight the syntax error.

```

SQL> select * from employees where dbkey = _dbkey'1,2,3';
%RDB-F-CONVERT_ERROR, invalid or unsupported data conversion
-RDMS-E-DBKFORMAT, database key format incorrect "1,?2,3" - unexpected
character
SQL> select * from employees where dbkey = _dbkey'-1:+2:0';
%RDB-F-CONVERT_ERROR, invalid or unsupported data conversion
-RDMS-E-DBKFORMAT, database key format incorrect "-1:+?2:0" - unexpected
character
SQL> select * from employees where dbkey = _dbkey'23:1:1';
%RDB-E-NO_RECORD, access by dbkey failed because dbkey is no longer associated
with a record
-RDMS-F-INVDBK, 23:1:1 is not a valid dbkey

```

7.6.6 RMU/SHOW STATISTIC "Logical Area" Statistics Consume Large Amounts of Memory

The RMU/SHOW STATISTIC utility consumes approximately 13 thousand bytes of virtual memory per logical area. Also, the number of logical areas is determined by the largest logical area identifier, not the actual number of areas.

This can result in the RMU/SHOW STATISTIC utility consuming large amounts of virtual memory, even if you do not wish to review logical area statistic information.

There currently is no method available to not display logical area statistic information.

This problem has been corrected in Oracle Rdb7 Release 7.0.2. A new qualifier for the RMU/SHOW STATISTICS command, /[NO]LOGICAL_AREA, can be used to indicate that you do not wish to display logical area statistic information. By specifying the /NOLOGICAL_AREA qualifier, the virtual memory for logical area statistic information presentation will not be acquired.

Be careful when specifying the /NOLOGICAL_AREA qualifier that you do not specify /NOLOG, which will cause logical area statistic information to still be collected.

The command default is /LOGICAL_AREA.

There is no corresponding configuration variable. This qualifier cannot be modified at run-time.

7.6.7 RMU/OPEN/STATISTICS, RMU/CLOSE/STATISTICS Enhancement

The database statistic information is lost when the database is closed. This makes it very difficult to accumulate statistic information for an extended period of time, such as a month or more, where the database is closed several times during that period.

There is no workaround to this problem.

This problem has been corrected in Oracle Rdb7 Release 7.0.2. The RMU/CLOSE command has been enhanced to include the /STATISTICS qualifier. The /STATISTICS qualifier indicates that the current statistic information should be preserved. The default is /NOSTATISTICS, which indicates statistic information is NOT preserved on database close, as is currently the case.

The RMU/OPEN command has also been enhanced to include the /STATISTICS qualifier. The /STATISTICS qualifier indicates that previously saved statistic information should be loaded when the database is opened. The default is /NOSTATISTICS, which indicates statistic information is NOT loaded on database open, as is currently the case.

The statistic information is stored in a node-specific database file, located in the same directory as the database root file. The file is named "database_node.RDS". For example, the MF_PERSONNEL database open on node MYNODE would use a statistic file named MF_PERSONNEL_MYNODE.RDS.

Note that clusterwide statistic information is NOT stored in the statistic file. This allows you to decide on which nodes the statistic information should be initially loaded at database open time.

The statistic files may not be renamed or copied; they contain node specific information. The statistic files may be deleted, if they are no longer needed. They are not required in order to open a database.

The statistic files will not be loaded if the physical schema of the database has changed since the statistic file was created. This means that the addition or deletion of storage areas, logical areas (tables and/or indexes) and record caches will invalidate the statistic files. This restriction prevents incorrect statistic information from being loaded when intervening physical changes occurred to the database. Closing the database will update the statistic files and make them valid again.

The /STATISTICS qualifier cannot be specified in conjunction with the RMU/CLOSE/CLUSTER command. To preserve the statistics information for a database open on a cluster, you must specifically close the individual nodes.

After the database is opened with the RMU/OPEN/STATISTICS command, the saved statistics file is closed. The statistics file is never automatically deleted. It is up to the user to manage the statistic files. If, at a later time, a user tries to RMU/OPEN a database with the /STATISTICS qualifier and the statistics file no longer exists, the database will still open fine. The error is logged in the monitor logfile.

The RMU/BACKUP command will not save the statistics files. They are considered temporary files and not considered part of the database.

RMU/SHOW USERS reports if the statistic information file was imported and whether or not the import failed.

If you specified the /STATISTICS qualifier on the RMU/OPEN command then the statistics information will be automatically preserved in the event of abnormal database closure (such as DBR failure). In order to ensure that the ondisk statistic information files are relatively accurate in the case of a node failure or monitor failure, the statistic information files are "checkpointed" by the database monitor every half-hour.

The RMU/SHOW USERS utility identifies when each database's checkpoint will occur as in the following example.

```
ALL> rmu/show users
Oracle Rdb X7.0-00 on node ALPHA3  4-NOV-1998 10:45:37.87
  - monitor started  4-NOV-1998 10:45:24.52 (uptime 0 00:00:13)
  - monitor log filename is "$111$DUA366:[RDMMON_LOGS]RDMMON711_ALPHA3.LOG;173
"
database _$111$DUA347:[R_ANDERSON.WORK.ALS]MF_PERSONNEL.RDB;1
  - first opened  4-NOV-1998 10:45:35.09 (elapsed 0 00:00:02)
  * database is opened by an operator
  * statistic information imported
  * next statistic information checkpoint at  4-NOV-1998 11:15:35.90
  - current after-image journal file is KODA_TEST:[R_ANDERSON.TMP]RICK1.AIJ;1
ALL> show time
4-NOV-1998 10:45:43
```

7.6.8 RCS Periodic Cache Validation

The Record Cache Server (RCS) process can periodically perform simple cache sanity and validity checks on all row caches. Areas that are checked include:

- Various row cache data structure contents
- TSN values of cached rows
- Hash chain contents

In order to reduce impact to the users of the caches, minimal locking of the caches is performed. This can, however, lead to false detections of problems (especially in the hash chain validations).

Certain 'serious' cache corruptions cause the RCS process to bugcheck (and thus the database to be closed). Lesser problems cause debug information to be written to the RCS log file.

To enable this feature, define the system logical name `RDM$BIND_RCS_VALIDATE_SECS` to the number of seconds between each cache validation pass. A value in the range of 300 (5 minutes) to 86400 (24 hours) is suggested. A value of 0 disables the cache validations. Once initiated, the interval can be re-set by changing the logical name definition; the logical is translated at each validation.

7.6.9 Hot Standby Support for Alternate Remote Nodename

The Hot Standby product has been enhanced to support an alternate remote nodename which identifies an available secondary network link to the standby database.

The purpose of the alternate remote nodename is to provide the master database with un-interrupted hot standby replication in case of network failure where multiple network links are available. Following network failure, the master database will automatically attempt to re-connect to the standby database using the alternate remote nodename information, if available.

The master database `RMU/REPLICATE AFTER_JOURNAL` command has been enhanced to support a new optional qualifier: `/ALT_REMOTE_NODE=nodename`. The `/ALT_REMOTE_NODE` qualifier can only be used in conjunction with the `/STANDBY_ROOT` qualifier, which specifies the standby database nodename.

The `/ALT_REMOTE_NODE` qualifier value identifies the alternate remote nodename of the standby database. The maximum length of the nodename is 31 characters. The alternate nodename can be the same as the nodename specified in the `/STANDBY_ROOT` qualifier. The nodename specified by the `/ALT_REMOTE_NODE` qualifier MUST identify the same standby database on the same remote node as originally specified using the `/STANDBY_ROOT` qualifier.

If the `/ALT_REMOTE_NODE` qualifier is not specified, the master database will automatically attempt to re-connect to the standby database using the original remote nodename specified using the `/STANDBY_ROOT` qualifier.

The `RMU/REPLICATE AFTER_JOURNAL CONFIGURE` command can also be used to store the alternate remote nodename in the database. As with the `START` command, the `/ALT_REMOTE_NODE` qualifier is specified in conjunction with the `/STANDBY_ROOT` qualifier.

The `RMU/REPLICATE AFTER_JOURNAL RESET` command will clear any previously configured alternate remote nodename information.

At run-time, the RDM\$BIND_HOT_NETWORK_ALT_NODE logical name can be defined in the LNM\$SYSTEM_TABLE table to override any alternate remote nodename information specified at hot standby startup. The logical must be specified on all nodes where the master database is open.

7.7 Enhancements Provided in Oracle Rdb7 Release 7.0.1.3

7.7.1 Exceeding Complex Query Limit Generated %RDMS-F-MAX_CCTX Error

Prior to Oracle Rdb Release 6.0, you could generate a complex query that exceeded the limit of 32 contexts. However, when more than 32 contexts were encountered for a single query, Oracle Rdb generated the following error:

```
%RDMS-F-MAX_CCTX exceeded maximum allowable context number
```

Examples of objects in a query that would count as a context are table references, views, inner selects, or aggregates.

For Oracle Rdb Release 6.0 and later releases, the context limit was raised from 32 contexts to 128 contexts.

7.7.2 New Maximum Equivalent Class Limit for Complex Queries

Bugs 611733 and 610614.

When a query uses many nested subqueries with equality predicates, the optimizer could reach its limit of equivalent classes. At that point, the query becomes very unpredictable, and finally runs out of memory.

Oracle Rdb7 optimizer has been enhanced to increase the maximum number of equivalent classes to 1024.

7.7.3 Monitor Consumes Less Virtual Memory when Opening Databases with Global Buffers

On large systems with very large numbers of global buffers, the Oracle Rdb7 monitor process (RDMMON) could have all of its process virtual address space consumed by a very small number of databases due to the amount of virtual address space needed to map the database global section. This could prevent additional databases from being opened on the node.

In order to help relieve this virtual memory limitation of the monitor process, the global buffers portion of the database global section is no longer mapped by the monitor. This global buffers portion of the database global section is generally the largest single portion of the global section and not mapping it can greatly reduce the amount of the monitor's virtual memory consumed by the database global section. For some databases, the amount of virtual memory that the monitor requires can be a small fraction of the total database global section size.

For example, a database with 20,000 global buffers and a buffer size of 6 blocks requires 120,000 pages (Alpha pagelets) of virtual address space for the global buffers themselves. The size of the entire database global section as shown by RMU/DUMP/HEADER is 70062212 bytes (136,840 pages):

```

$ RMU/DUMP/HEADER DKA0:[DB]MYDB.RDB;1
.
.
.
Derived Data...
- Global section size
  With global buffers disabled is 379982 bytes
  With global buffers enabled is 70062212 bytes

```

Because the global buffers are not mapped, the monitor process only maps 16,893 of the 136,840 pages for a savings of 120,000 pages of virtual memory. This savings can allow the monitor to keep more databases concurrently open before its process virtual address space would be consumed.

The following example shows a portion of the monitor log file for a database open request for a database with 11,000 global buffers. The size of the database global section is 75,631 pages but the monitor process maps only 9,631 into virtual memory.

```

16-Mar-1998 02:56:18.92 - received open database request from 22E00479:0
- process name random@TNA23, user RDENT
- for database "_$1$DIA0:[DB]MYDB.RDB;1" [_$1$DIA0] (271,893,0)
- number of global buffers is 11000, maximum buffers per user is 5
- database global section name is "RDM70T_8K1ADR00"
- database global section size is 75631 pages (512 bytes per page)
- monitor maps 9631 pages of the global section into virtual memory

```

User processes continue to map the entire database global section, it is only the monitor process that does not map the global buffers portion of the global section.

7.7.4 Restrictions Lifted for Strict Partitioning

Bug 548039.

When a table's storage map has the attribute PARTITIONING IS NOT UPDATABLE, mapping of data to a storage area is strictly enforced. This is known as **strict partitioning**. This release of Oracle Rdb7 lifts restrictions imposed by earlier releases as described below.

- Strict partitioning now enforced at runtime.

In prior releases of Oracle Rdb7, the PARTITIONING IS NOT UPDATABLE rule was enforced during query compilation. Therefore, any UPDATE statement, procedure, or trigger definition which attempted to update the partitioning columns were rejected. This created a problem for 4GL tools and generated applications which didn't know that these columns were not allowed to appear in an UPDATE statement.

This enforcement has been lessened for this release. The enforcement is now data-value based and allows updates to these columns if the data values do not change. The prior data values are compared with the new row/column values and any changes are reported as runtime errors. If no rows are updated or the column values do not change, then the update is permitted. This allows 4GL tools and generated applications to reference these columns in a generalized UPDATE statement.

Note

A small amount of CPU time overhead is added to the UPDATE statement which must save and compare the partitioning column values. If an

UPDATE statement avoids referencing these columns then this overhead is eliminated.

- **Locking behavior for ISOLATION SERIALIZABLE**

In prior releases of Oracle Rdb7 when the current transaction is started using the ISOLATION SERIALIZABLE level (the default), all partitions of a table are locked in protected mode. This was done to enforce the serializable characteristic of the transaction.

However, if a strictly partitioned query is being performed, not all the partitions need to be locked so strongly. The serializable characteristic of the transaction can be guaranteed by only locking the partitions used by the query.

In this release of Oracle Rdb7, each partition is locked when it is referenced, and therefore concurrent sequential access to a strictly partitioned table is now possible. If the application needs to have partitions locked immediately rather than as they are referenced, or in a stronger exclusive mode, then the SET TRANSACTION .. RESERVING PARTITION clause should be used (see Section 7.7.16).

7.7.5 Date Subtraction

Some Oracle applications rely on being able to subtract one date from another and getting back the number of days between the two dates. In an effort to better support those applications, that support has been provided in the Oracle Level1 dialect.

Unlike Oracle, however, partial days are not returned. The result is always an integer value.

The following example shows the subtraction of dates:

```
SQL> SET DIALECT 'ORACLE LEVEL1';
SQL> SELECT SYSDATE - DATE VMS '12-JAN-1998' FROM RDB$DATABASE;

          15
1 row selected
SQL>
```

7.7.6 Default Node Size Now Displayed After Index Is Created

In prior releases of Oracle Rdb7, a CREATE INDEX statement would supply a default index node size if none were provided for a UNIQUE SORTED index, or a SORTED RANKED index. However, neither the SQL SHOW INDEX, SHOW TABLE nor RMU/EXTRACT statements would display the value of this default node size.

This problem has been corrected in Oracle Rdb7 Version 7.0.1.3. All new indexes will have stored the default node size for display by SQL and RMU/EXTRACT statements.

The following example the default node size is displayed after an index is created.

```

SQL> -- Create a simple table upon which we can define
SQL> -- some indices
SQL>
SQL> CREATE TABLE TEST_INDEX_TABLE
cont>   (A CHAR(70),
cont>   B INTEGER);
SQL>
SQL> -- Default value is 430 bytes
SQL>
SQL> CREATE UNIQUE INDEX TEST_INDEX_DEF
cont>   ON TEST_INDEX_TABLE (A, B)
cont>   TYPE IS SORTED
cont>   USAGE UPDATE;
SQL>
SQL> SHOW TABLE (INDEX) TEST_INDEX_TABLE
Information for table TEST_INDEX_TABLE

TEST_INDEX_DEF                with column A
                               and column B

  No Duplicates allowed
  Type is Sorted
  Compression is DISABLED
  Node size 430
  Percent fill 70

```

7.7.7 RUJ Buffers in a Global Section When Row Cache is Enabled

For row caches, recovery unit journaling (RUJ) must logically come before each modification to any record residing in a row cache. Having the RUJ information is critical in returning the row to its before-image state in the event that the modifying transaction rolls back or aborts abnormally. To minimize the occurrences of these synchronous RUJ I/Os, Oracle Rdb defers for as long as possible the writing of modified records into the row cache. The synchronous I/O includes all updated rows since the previous RUJ I/O.

If an application performs a large number of inserts or updates to a table contained in a row cache, a high number of these RUJ I/Os may be seen. To eliminate the majority of these RUJ I/Os, a system logical name, `RDM$BIND_RUJ_GLOBAL_SECTION_ENABLED`, has been added that you can use to specify whether you want the before-image records to be written to process-private memory (the traditional method) or to a system-wide, shared memory, global section.

When the global section option is chosen, the RUJ information is made available to any possible future database recovery process from the shared memory global section. Traditionally, such information was only shared by writing the information to the RUJ file which the DBR process could read. By adding this capability, only an in-memory I/O is required before modifying a row in the row cache.

When a process terminates abnormally, Oracle Rdb activates a database recovery (DBR) process to recover the work done by the terminated user. The DBR process performs an "undo" operation, or rollback, of the process' outstanding, uncommitted transactions, if any. If the system-wide DBR process buffers are enabled, the DBR process first writes the current RUJ buffer to the RUJ file. It then recovers the RUJ file placing the before-image of each record back on the database page. If the DBKEY for that record is also found in a row cache, the before-image is placed back into the row cache as well.

To enable this optimization, define the logical name `RDM$BIND_RUJ_GLOBAL_SECTION_ENABLED` to "1" in the system logical name table. The global section created for the RUJ buffers will be about 256 VAX pages or 16 Alpha Pages for each allowed user of a database. One global section will be created for each database that has row cache enabled. Databases that do not have row cache enabled will not have the RUJ global buffer optimization enabled.

The following OpenVMS system parameters will also need to be modified:

- `GBLSECTIONS` will need to be increased by the maximum number of Oracle Rdb databases open at one time on the system.
- `GBLPAGES` will need to be increased by 256 times the maximum number of users for each databases open at one time on the system.
- `GBLPAGFIL` will need to be increased by either 256 (on OpenVMS VAX) or 16 (on OpenVMS Alpha) times the maximum number of users for each databases open at one time on the system.

There is no additional virtual memory consumption for databases users when the RUJ global buffers optimization is enabled; each user process continues to use the same amount of virtual memory (256 blocks) as when the optimization is not enabled.

7.7.8 Enhancements to Range Queries on SORTED Indexes

Bug 500856.

In previous versions of Oracle Rdb, the last index key fetched from the index partition during a range query was used to determine if the scan was complete for the current range or if the next partition needed to be scanned. This could result in unnecessary scans of subsequent index partitions if the last fetched value in the SORTED index partition was not beyond the query range.

There are two important benefits to this enhancement. First, there is a reduction in I/O because fewer storage areas need to be accessed. Second, because there is no need to access subsequent partitions, there are now a smaller number of index partitions locked, thus allowing more concurrency. In cases where the next partition is empty, it is possible for more than one partition to be scanned and locked.

Note: Some users may see no change in behavior because the last key value in the index partition may have been beyond the query bounds or, in the case of a unique index definition with an exact match query, a direct key lookup may result as shown below.

```
SQL> SELECT COUNT(*) FROM EMPLOYEES WHERE EMPLOYEE_ID = '00200';
Aggregate      Index only retrieval of relation EMPLOYEES
  Index name  IDX1 [1:1]          Direct lookup
```

The following example shows a partitioned index and three queries. Each query is run in a different process and attaches to the same database.

In previous releases of Oracle Rdb, the first query would lock AREA1 and AREA2 when it only required scanning of AREA1. The second query would then lock AREA2 and AREA_OTHER when it only required scanning of AREA2. Thus, the three queries could not execute concurrently.

The following example demonstrates that a smaller number of index partitions are locked:

```
SQL> CREATE INDEX EMP_INDEX ON EMPLOYEES (EMPLOYEE_ID)
cont> TYPE IS SORTED
cont> STORE USING (EMPLOYEE_ID)
cont>     IN AREA1 WITH LIMIT OF ('00200')
cont>     IN AREA2 WITH LIMIT OF ('00400')
cont>     OTHERWISE IN AREA_OTHER;
SQL>
SQL> -- This query previously locked AREA1 and AREA2.
SQL> -- With the new algorithm, only AREA1 is locked.
SQL>
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID < ('00199');
6 rows deleted
SQL>
SQL> -- This query previously locked AREA2 and AREA_OTHER
SQL> -- With the new algorithm, only AREA2 is locked.
SQL>
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID > ('00201') AND
cont> EMPLOYEE_ID < ('00399');
5 rows deleted
SQL> -- This query locks AREA_OTHER
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID > ('00401');
23 rows deleted
```

The following example demonstrates fewer areas scanned with the new algorithm resulting in less I/O:

```
SQL> CREATE INDEX INDEX_EMP
cont>     ON EMPLOYEES (EMPLOYEE_ID)
cont>     TYPE IS SORTED
cont>     STORE
cont>     USING (EMPLOYEE_ID)
cont>     IN UNIFORM1
cont>     WITH LIMIT OF ('00100')
cont>     IN UNIFORM2
cont>     WITH LIMIT OF ('00200')
cont>     IN UNIFORM3
cont>     WITH LIMIT OF ('00300')
cont>     OTHERWISE IN UNIFORM4;
SQL>
SQL> -- First, delete all employees records in UNIFORM1, UNIFORM2, UNIFORM3
SQL>
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID BETWEEN '00001' AND '00300';
12 rows deleted
SQL>
SQL>
SQL> -- Previously, the following query would result in reading from areas
SQL> -- UNIFORM1, UNIFORM2, UNIFORM3, and UNIFORM4. This occurred because
SQL> -- all partitions were scanned until an index key was found to end the scan.
SQL> -- With the new algorithm, only UNIFORM1 is read, resulting in less I/O.
SQL>
SQL> -- By turning on debug flags (STRATEGY, EXECUTION, INDEX_PARTITION),
SQL> -- the index partitions scanned are displayed.
SQL>
SQL> SET FLAGS 'STRATEGY,EXECUTION,INDEX_PARTITION';
SQL> SELECT * FROM EMPLOYEES WHERE EMPLOYEE_ID = '00020';
~S#0004
Leaf#01 Ffirst EMPLOYEES Card=40
  BgrNdx1 INDEX_EMP [1:1] Fan=17
~E#0004.2 Start Area INDEX_EMP.UNIFORM1 (1) <--- ** index partition scanned **
~E#0004.01(1) BgrNdx1 EofData DBKeys=0 Fetches=0+0 RecsOut=0 #Bufs=0
0 rows selected
```

The same query in previous versions of Rdb7, would result in the empty index partitions being scanned until an index key was found to end the scan.

```
SQL> SET FLAGS 'STRATEGY,EXECUTION,INDEX_PARTITION';
SQL> SELECT * FROM EMPLOYEES WHERE EMPLOYEE_ID = '00020';
~S#0002
Leaf#01 FFirst EMPLOYEES Card=40
  BgrNdx1 INDEX_EMP [1:1] Fan=17
~E#0002.1 Start Area IDX1.UNIFORM1 (1) <--- ** index partitions scanned **
~E#0002.1 Next Area IDX1.UNIFORM2 (2)
~E#0002.1 Next Area IDX1.UNIFORM3 (3)
~E#0002.1 Next Area IDX1.UNIFORM3 (4)
0 rows selected
```

The new algorithm utilizes other data structures to determine that all the data has been returned for the query and eliminates unnecessary index area scans based on the index partition values.

Note

In order to utilize the new index partition scanning algorithm, the logical name RDMS\$INDEX_PART_CHECK must be defined to 1. Otherwise, the default is to use the old scanning behavior for partitioned indexes (the same as defining RDMS\$INDEX_PART_CHECK = 0 or not defining the logical at all).

This index partition enhancement is not supported for mapped indexes or descending indexes.

7.7.9 UPDATE STATISTICS Clause for ALTER TABLE Statement Implemented for /TYPE=NREL

It is now possible to reacquire table statistics when the ATTACH type is NREL (non-relational DBI gateways). This is done by using the ALTER TABLE UPDATE STATISTICS statement. In prior versions, this was only allowed when the ATTACH type was DBI.

Use of this statement will update the table cardinality and may improve optimization strategies. For example,

```
SQL> ATTACH 'FILE /TYPE=NREL/PATH=PATH-NAME/DICT=DICTIONARY-DRIVER-NAME' ;
SQL> SELECT RDB$CARDINALITY FROM RDB$RELATIONS
cont> WHERE RDB$RELATION_NAME = 'table-name' ;
  RDB$CARDINALITY
  0
1 row selected
SQL> ALTER TABLE table-name UPDATE STATISTICS ;
SQL> SELECT RDB$CARDINALITY FROM RDB$RELATIONS
cont> WHERE RDB$RELATION_NAME = 'table-name' ;
  RDB$CARDINALITY
  322
1 row selected
```

This problem has been corrected in Oracle Rdb7 Version 7.0.1.3.

7.7.10 Relaxed Privilege Checking for Temporary Tables

In prior versions of Oracle Rdb7, privileges required for data manipulation operations on global and local temporary tables were the same as those required for base tables. For example, to perform an insert into a global temporary table, a user needed SELECT and INSERT privileges at the database level.

This requirement existed because an insert into a base table implicitly inserts data into the database. The privilege granted at the database level was used to filter the privileges for the table.

However, unlike base tables, the data in temporary tables is not actually stored in the database, thus temporary tables never update the database.

In this release of Oracle Rdb7, only the privileges associated with the temporary table will be considered when performing security validation during data manipulation operations. For example, if the user can attach to the database (requires SELECT privilege only) and is granted INSERT to a global or local temporary table, then the user (or an invokers rights stored routine) will be permitted to update the temporary table. This change will affect the operation of SQL*net for Rdb which no longer requires database manipulation privileges (INSERT, UPDATE, DELETE) for processing temporary tables.

Note

This is a relaxation of the security checking from prior versions of Oracle Rdb7 and only applies to temporary tables.

For previous versions, definers rights stored procedures could be utilized to access the temporary table. The DECLARE LOCAL TEMPORARY TABLE clause generates a "scratch" temporary table which has no associated access control. It is managed by the module which declares it. This type of temporary table is also available through dynamic SQL. This change has been implemented in Oracle Rdb7 Release 7.0.1.3.

7.7.11 Improved Estimation for INDEX Column References

The Oracle Rdb optimizer always seems to estimate much higher cardinalities for the chosen solution when the selection predicate specifies only some of the leading segments on a multisegment index. For instance, if you specify an equality on the first segment of a two segment index.

In the past, this slight overestimation was not a significant problem on relatively small tables but becomes a more significant problem when the select involves a sort (in particular the OpenVMS SORT facility) where the sort buffer is pre-allocated based on its estimated cardinality of the solution.

In the following example, the table STUDENTS has an index on the two columns STU_NUM and COURSE_NUM. The optimizer uses a fixed proportion of the table cardinality based on the equality with the STU_NUM column and 5134 rows are expected when, in reality, only 9 are returned by the query.

```

SQL> CREATE INDEX STUDENT_NDX ON STUDENTS (STU_NUM,COURSE_NUM DESC);
SQL>
SQL> SELECT STU_NUM FROM STUDENTS
cont> WHERE STU_NUM = 191270771
cont> ORDER BY OTHER_COLUMN;
Solutions tried 2
Solutions blocks created 1
Created solutions pruned 0
Cost of the chosen solution 4.5644922E+03
Cardinality of chosen solution 5.1342500E+03
~O: Physical statistics used
Sort
SortId# 7., # Keys 2
  Item# 1, Dtype: 2, Order: 0, Off: 0, Len: 1
  Item# 2, Dtype: 35, Order: 0, Off: 1, Len: 8
  LRL: 32, NoDups:0, Blks:327, EqlKey:0, WkFls: 2
Leaf#01 BgrOnly STUDENTS Card=164296
  BgrNdx1 STUDENT_NDX [1:1] Fan=14
  191270771
  191270771
  191270771
  191270771
  191270771
  191270771
  191270771
  191270771
  191270771
  191270771
SORT(9) SortId# 7, ----- Version: V5-000
Records Input: 9      Sorted: 9      Output: 0
LogRecLen Input: 32  Intern: 32      Output: 32
Nodes in SoTree: 5234  Init Dispersion Runs: 0
Max Merge Order: 0    Numb.of Merge passes: 0
Work File Alloc: 0
MBC for Input: 0      MBC for Output: 0
MBF for Input: 0      MBF for Output: 0
Big Allocated Chunk: 4606464 busy
  191270771
9 rows selected

```

Starting with this release of Oracle Rdb, the SET FLAGS command (and the companion logical name RDMS\$SET_FLAGS) allow applications to make better use of the index column group information specified in the indices.

```
SQL> SET FLAGS 'INDEX_COLUMN_GROUP';
```

This will activate the optimizer to consider the index segment columns as workload column group, compute the statistics for duplicity factor and null factor on the fly, and apply them in estimating the cardinality of the solution.

The following is the optimizer cost estimate and sort output trace when the user enables this feature. In this example the optimizer estimates a lower cardinality of about 8 rows.

```

Solutions tried 2
Solutions blocks created 1
Created solutions pruned 0
Cost of the chosen solution 3.8118614E+01
Cardinality of chosen solution 8.3961573E+00
~O: Workload and Physical statistics used
Sort
SortId# 2., # Keys 2
  Item# 1, Dtype: 2, Order: 0, Off: 0, Len: 1
  Item# 2, Dtype: 35, Order: 0, Off: 1, Len: 8
  LRL: 32, NoDups:0, Blks:7, EqlKey:0, WkFls: 2
Leaf#01 BgrOnly STUDENTS Card=164296
  BgrNdx1 STUDENT_NDX [1:1] Fan=14
  191270771
  191270771
  191270771
  191270771
  191270771
  191270771
  191270771
  191270771
  191270771
SORT(2) SortId# 2, ----- Version: V5-000
  Records Input: 9      Sorted: 9      Output: 0
  LogRecLen Input: 32  Intern: 32      Output: 32
  Nodes in SoTree: 114  Init Dispersion Runs: 0
  Max Merge Order: 0    Numb.of Merge passes: 0
  Work File Alloc: 0
  MBC for Input: 0      MBC for Output: 0
  MBF for Input: 0      MBF for Output: 0
  Big Allocated Chunk: 87552 idle
  191270771
  9 rows selected

```

For prior releases of Rdb, the database administrator can collect workload statistics using RMU/COLLECT OPTIMIZER.

This change is available in Oracle Rdb7 Version 7.0.1.3. In a future release the use of INDEX_COLUMN_GROUP will become the default behavior when using the new Rdb costing model.

7.7.12 SQL92 Intermediate Level UNIQUE Constraint Available in Rdb7

Oracle Rdb now provides an SQL92 intermediate level compliant UNIQUE constraint. This type of constraint excludes columns which are NULL from the UNIQUE comparison. This effectively allows sets of columns to be UNIQUE or NULL.

This type of constraint will be created by default when the SQL dialect is set to 'SQL89', 'MIA', 'ORACLE LEVEL1' or 'SQL92'. The default dialect is SQLV40. Oracle recommends that you set the dialect to SQL92 (or one of the listed dialects) before using CREATE or ALTER TABLE to add UNIQUE constraints to tables.

Note

The new UNIQUE semantics will be used at run-time under any selected dialect. But the table must be created or altered under the listed dialects to have the new style of unique enabled.

Improved Performance

In addition to conforming to the database language SQL standard (SQL92 Intermediate Level), the new UNIQUE constraint implementation also provides improved performance for single row inserts. This is made possible by eliminating checks for NULL values from the selection expression and thus simplifying the optimization for unique checking.

Here is a comparison of the old and new optimizer strategies. In this example a UNIQUE constraint ("UNIQUE_A") and index on column A is used to check for uniqueness during an INSERT statement. Note that the optimizer chooses a full range search of the index ([0:0]).

```
~S: Constraint "UNIQUE_A" evaluated
Cross block of 2 entries
  Cross block entry 1
    Conjunct      Firstn Get      Retrieval by DBK of relation T_UNIQUE
  Cross block entry 2
    Conjunct      Aggregate-F2    Conjunct
    Index only retrieval of relation T_UNIQUE
    Index name    T_UNIQUE_INDEX_A [0:0]
```

With the simplified UNIQUE constraint ("UNIQUE_B") the optimizer can use a direct lookup of the index ([1:1]) which reduces the I/O to the index when performing the constraint evaluation.

```
~S: Constraint "UNIQUE_B" evaluated
Cross block of 2 entries
  Cross block entry 1
    Conjunct      Firstn Get      Retrieval by DBK of relation T_UNIQUE
  Cross block entry 2
    Conjunct      Aggregate-F2    Index only retrieval of relation T_UNIQUE
    Index name    T_UNIQUE_INDEX_B [1:1]
```

Upward Compatibility

In prior versions, the UNIQUE constraint restricted columns to a single NULL value. If you wish to retain this behavior, use the SET DIALECT 'SQLV40' statement before creating new tables or altering existing tables to add UNIQUE constraints.

UNIQUE constraints created in previous versions of Oracle Rdb will still perform as expected. Interfaces such as RDO or the CDD/Repository will continue to define the older style UNIQUE constraint. It is expected that future versions of the Oracle CDD/Repository will implement the new UNIQUE constraint. Database EXPORT and IMPORT will retain the UNIQUE constraint characteristics as defined by the database administrator, regardless of the defined dialect setting.

Note

RMU Extract Item=Table will not distinguish between the old and new UNIQUE constraints in this release of Rdb. The generated SQL script must be modified to establish the appropriate dialect before using it to create a database.

Because this new style of UNIQUE constraint is a relaxation of the UNIQUE rules, it is possible to drop the old style UNIQUE constraint and redefine the constraint under the SQL92 dialect.

Note that this meaning of UNIQUE (excluding NULL from the uniqueness test) does not apply to the UNIQUE index which still does not allow duplicate entries for NULL. If a UNIQUE index is currently defined which assists the UNIQUE constraint optimization, the database administrator may wish to drop the index and make it a non-UNIQUE index so that multiple NULLs can be stored. The UNIQUE constraint will still enforce the uniqueness of the data.

You can use the SQL SHOW TABLE command to determine which type of UNIQUE constraint is in use. The following example shows a UNIQUE constraint created when the default dialect was used (SQLV40). A new description follows the "Unique constraint" text, explaining the interpretation of null values.

```
SQL> SHOW TABLE (CONSTRAINT) T_UNIQUE
Information for table T_UNIQUE

Table constraints for T_UNIQUE:
T_UNIQUE_UNIQUE_B_A
Unique constraint
  Null values are considered the same
Table constraint for T_UNIQUE
Evaluated on UPDATE, NOT DEFERRABLE
Source:
  UNIQUE (b,a)
.
.
.
```

The next example shows a UNIQUE constraint created when the dialect was set to 'SQL92', and the description here indicates that all null values are considered distinct.

```
SQL> SHOW TABLE (CONSTRAINT) T_UNIQUE2
Information for table T_UNIQUE2

Table constraints for T_UNIQUE2:
T_UNIQUE2_UNIQUE_B_A
Unique constraint
  Null values are considered distinct
Table constraint for T_UNIQUE2
Evaluated on UPDATE, NOT DEFERRABLE
Source:
  UNIQUE (b,a)
.
.
.
```

Additional Constraint Improvements

As a side effect of this change, Oracle Rdb also recognizes a larger class of CHECK constraints as being uniqueness checks. The main benefit is that these constraints are no longer processed when a DELETE is executed for the table, because DELETE does not affect the uniqueness of the remaining rows.

The following is an example of this CHECK constraint:

```

SQL> CREATE TABLE T_USER_UNIQUE_NEW (
cont>     A INTEGER,
cont>     B INTEGER,
cont>     CONSTRAINT UNIQUE_AB_NEW
cont>     CHECK ((SELECT COUNT(*)
cont>             FROM T_USER_UNIQUE_NEW t2
cont>             WHERE T2.A = T_USER_UNIQUE_NEW.A AND
cont>                 T2.B = T_USER_UNIQUE_NEW.B) <= 1)
cont>     NOT DEFERRABLE
cont> );

```

In previous versions of Rdb only equality with 1 was recognized as a uniqueness constraint. In this example, a comparison of LESS THAN or EQUAL to one also qualifies as a uniqueness constraint.

7.7.13 Enhancements to DROP STORAGE AREA ... CASCADE

Oracle Rdb7 Release 7.0.1.3 contains several corrections and enhancements to the DROP STORAGE AREA ... CASCADE feature.

DROP INDEX ... CASCADE is Performed if Whole Index is in a Single Area

In previous releases, the DROP STORAGE AREA ... CASCADE command would fail if a partitioned table had an index which was not partitioned and it resided entirely in the storage area being dropped.

This restriction has been removed. Now the index itself will be dropped using CASCADE semantics and this will invalidate any query outlines that reference the index.

Not all Constraints are Evaluated by DROP STORAGE AREA ... CASCADE

The NOT NULL, PRIMARY KEY, and UNIQUE constraints for affected tables are ignored by DROP STORAGE AREA ... CASCADE in this release, because validation of these constraints is not warranted.

These types of constraints are not affected by the removal of rows from a table. This can save considerable I/O and elapsed time when performing the DROP STORAGE AREA ... CASCADE command. However, CHECK and FOREIGN constraints on the affected table, and referencing tables, will still be evaluated.

Debugging Output now Available for DROP STORAGE AREA ... CASCADE

When the DROP STORAGE AREA ... CASCADE command is executing, it will log debugging messages to the standard output device (SYSS\$OUTPUT) or the RDMS\$DEBUG_FLAGS_OUTPUT log file, if defined.

This logging can be enabled using the new logical name RDMS\$SET_FLAGS which accepts the same input as the SQL SET FLAGS statement.

```
$ DEFINE RDMS$SET_FLAGS 'STOMAP_STATS,INDEX_STATS,ITEM_LIST'
```

These SET FLAGS options enable the following debug output:

- STOMAP_STATS will display the processing of storage maps for any tables which reference the dropped storage area. The output will be prefixed with "~As". This is identical to using RDMS\$DEBUG_FLAGS defined as "As".
- INDEX_STATS will display the processing of any indexes which reference the dropped storage area. The output will be prefixed with "~Ai". This is identical to using RDMS\$DEBUG_FLAGS defined as "Ai".
- ITEM_LIST will display the names of any constraints that require processing. This is identical to using RDMS\$DEBUG_FLAGS defined as "H".

The output includes the discovered tables and indexes, some decision point information (does an index need to be deleted?, does a partition need to be scanned?), and I/O statistics for the storage map pruning operations.

As part of the DROP STORAGE AREA ... CASCADE operation, tables and indexes may be deleted. These are processed internally as DROP TABLE ... CASCADE and DROP INDEX ... CASCADE operations. However, by the time these commands execute, all references to the dropped storage area will have been removed so, in many cases, the DROP simply cleans up the metadata definition and need not scan the storage area.

In the following example it can be seen that a single DROP STORAGE AREA ... CASCADE operation needs to scan four logical areas to destroy the hash indexes (see "destroy hash" in the example). The scanning of an area takes I/O and time and should be avoided if possible.

```
SQL> ALTER DATABASE
cont>     FILENAME 'TEST_MFDB'
cont>     DROP STORAGE AREA S_AREA_1A CASCADE;
~As: Drop Storage Area "S_AREA_1A" Cascade
~As: ...area referenced by map: "SR_MAP"
~As: ...area referenced by map: "PV_MAP"
~As: ...area referenced by map: "S_MAP"
~As: ...area referenced by map: "SF_MAP"
~As: ...area referenced by index: "SR_1H"
~As: ...area referenced by index: "PV_2H"
~As: ...area referenced by index: "S_1H"
~As: ...area referenced by index: "SF_1H"
~As: ...update the AIP for larea=64 (table)
~As: ...update the AIP for larea=65 (table)
~As: ...update the AIP for larea=66 (table)
~As: ...update the AIP for larea=67 (table)
~As: ...update the AIP for larea=56 (index)
~As: ...update the AIP for larea=58 (index)
~As: ...update the AIP for larea=60 (index)
~As: ...update the AIP for larea=62 (index)
~As: ...update the AIP for larea=47 (sysrec)
~As: ...drop table "SF" cascade
~Ai delete index (cascade) SF_2H
    destroy Hash index, Idx=57, Sys=48
~Ai delete index (cascade) SF_1H
~As: ...drop table "S" cascade
~Ai delete index (cascade) S_4H
    destroy Hash index, Idx=59, Sys=50
~Ai delete index (cascade) S_1H
~As: ...drop table "PV" cascade
~Ai delete index (cascade) PV_4H
    destroy Hash index, Idx=61, Sys=51
~Ai delete index (cascade) PV_2H
~As: ...drop table "SR" cascade
~Ai delete index (cascade) SR_2H
    destroy Hash index, Idx=63, Sys=49
~Ai delete index (cascade) SR_1H
~As: ...4 logical areas were scanned in other areas
~As: ...Reads: async 477 synch 103, Writes: async 144 synch 22
```

This revised script drops several areas in a specific order so that no logical area scans are performed. Even for this simple example database, the read/write I/O statistics (on the last line of each log) can be compared to see the improvement.

```

SQL> ALTER DATABASE
cont>     FILENAME 'TEST_MFDB'
cont>     DROP STORAGE AREA SF_AREA_1A CASCADE
cont>     DROP STORAGE AREA S_AREA_4A CASCADE
cont>     DROP STORAGE AREA PV_AREA_4A CASCADE
cont>     DROP STORAGE AREA SR_AREA_1A CASCADE
cont>     DROP STORAGE AREA S_AREA_1A CASCADE;
~As: Drop Storage Area "SF_AREA_1A" Cascade
~As: ...area referenced by index: "SF_2H"
~As: ...dropping index "SF_2H" (not partitioned)
~As: ...update the AIP for larea=57 (index)
~As: ...update the AIP for larea=48 (sysrec)
~As: ...drop index "SF_2H" cascade
~Ai delete index SF_2H                (1)
~As: ...Reads: async 0 synch 15, Writes: async 11 synch 4
~As: Drop Storage Area "S_AREA_4A" Cascade
~As: ...area referenced by index: "S_4H"
~As: ...dropping index "S_4H" (not partitioned)
~As: ...update the AIP for larea=59 (index)
~As: ...update the AIP for larea=50 (sysrec)
~As: ...drop index "S_4H" cascade
~Ai delete index S_4H                (1)
~As: ...Reads: async 0 synch 1, Writes: async 0 synch 7
~As: Drop Storage Area "PV_AREA_4A" Cascade
~As: ...area referenced by index: "PV_4H"
~As: ...dropping index "PV_4H" (not partitioned)
~As: ...update the AIP for larea=61 (index)
~As: ...update the AIP for larea=51 (sysrec)
~As: ...drop index "PV_4H" cascade
~Ai delete index PV_4H                (1)
~As: ...Reads: async 0 synch 2, Writes: async 0 synch 17
~As: Drop Storage Area "SR_AREA_1A" Cascade
~As: ...area referenced by index: "SR_2H"
~As: ...dropping index "SR_2H" (not partitioned)
~As: ...update the AIP for larea=63 (index)
~As: ...update the AIP for larea=49 (sysrec)
~As: ...drop index "SR_2H" cascade
~Ai delete index SR_2H                (1)
~As: ...Reads: async 0 synch 0, Writes: async 0 synch 18
~As: Drop Storage Area "S_AREA_1A" Cascade
~As: ...area referenced by map: "SR_MAP"
~As: ...area referenced by map: "PV_MAP"
~As: ...area referenced by map: "S_MAP"
~As: ...area referenced by map: "SF_MAP"
~As: ...area referenced by index: "SR_1H"
~As: ...area referenced by index: "PV_2H"
~As: ...area referenced by index: "S_1H"
~As: ...area referenced by index: "SF_1H"
~As: ...update the AIP for larea=64 (table)
~As: ...update the AIP for larea=65 (table)
~As: ...update the AIP for larea=66 (table)
~As: ...update the AIP for larea=67 (table)
~As: ...update the AIP for larea=56 (index)
~As: ...update the AIP for larea=58 (index)
~As: ...update the AIP for larea=60 (index)
~As: ...update the AIP for larea=62 (index)
~As: ...update the AIP for larea=47 (sysrec)
~As: ...drop table "SF" cascade
~Ai delete index (cascade) SF_1H
~As: ...drop table "S" cascade
~Ai delete index (cascade) S_1H
~As: ...drop table "PV" cascade
~Ai delete index (cascade) PV_2H
~As: ...drop table "SR" cascade
~Ai delete index (cascade) SR_1H

```

~As: ...Reads: async 0 synch 55, Writes: async 96 synch 32

The time it takes to delete the storage area file will depend on the size of the directory file, the file allocation, and also the number of extents made by the file system to expand the file. If the ERASE ON DELETE attribute is enabled on the disk, this must also be factored into the time calculations (allow time for the file system to overwrite the file with an erase pattern).

Note that the read/write I/O statistics are only output if the database has statistics collection enabled. Statistics collection may be disabled when the logical name RDM\$BIND_STATS_ENABLED is assigned the value 0, or in the database using the ALTER DATABASE ... STATISTICS COLLECTION IS DISABLED command.

7.7.14 New SQL SET FLAGS Options

New keywords for the SET FLAGS statement

This release of Oracle Rdb7 adds new keywords for use by the SET FLAGS statement and the RDM\$SET_FLAGS logical name. The keywords are not case sensitive and can be abbreviated to any unambiguous prefix.

Table 7-3 Rdb Flag Keywords

Keyword	Negated Keyword	Debug Flags Equivalent	Comment
COSTING	NOCOSTING ¹	Oc	Displays traces on optimizer costing
CURSOR_STATS	NOCURSOR_STATUS ¹	Og	Displays general cursor statistics for optimizer
INDEX_COLUMN_GROUP	NOINDEX_COLUMN_GROUP ¹	n/a	Enables leading index columns as workload column group. This may increase solution cardinality accuracy
SOLUTIONS	NOSOLUTIONS ¹	Os	Displays traces on optimizer solutions
TRANSITIVITY ¹	NOTRANSITIVITY	RDM\$DISABLE_TRANSITIVITY	Enables transitivity between selections and join predicates
MAX_STABILITY	NOMAX_STABILITY ¹	RDM\$MAX_STABILITY	Enables maximum stability (dynamic optimizer not allowed)
OLD_COST_MODEL	NOOLD_COST_MODEL ¹	RDM\$USE_OLD_COST_MODEL	Enables old cost model
REVERSE_SCAN ¹	NOREVERSE_SCAN	RDM\$DISABLE_REVERSE_SCAN	Enables reverse index scan strategy.
ZIGZAG_MATCH ¹	NOZIGZAG_MATCH	RDM\$DISABLE_ZIGZAG_MATCH	Enables zigzag key skip on both outer and inner match loops. ²

¹Default value

²ZIGZAG_MATCH, NOZIGZAG_OUTER disables zigzag key skip on outer loop (equivalent to defining the logical name RDM\$DISABLE_ZIGZAG_MATCH to a value of 1). NOZIGZAG_MATCH disables zigzag key skip on both outer and inner match loops (equivalent to defining the logical name RDM\$DISABLE_ZIGZAG_MATCH to a value of 2)

(continued on next page)

Table 7–3 (Cont.) Rdb Flag Keywords

Keyword	Negated Keyword	Debug Flags Equivalent	Comment
ZIGZAG_OUTER ¹	NOZIGZAG_OUTER	RDMSS\$DISABLE_ZIGZAG_MATCH	Enables zigzag key skip on outer loop. ²

¹Default value

²ZIGZAG_MATCH, NOZIGZAG_OUTER disables zigzag key skip on outer loop (equivalent to defining the logical name RDMSS\$DISABLE_ZIGZAG_MATCH to a value of 1). NOZIGZAG_MATCH disables zigzag key skip on both outer and inner match loops (equivalent to defining the logical name RDMSS\$DISABLE_ZIGZAG_MATCH to a value of 2)

New logical name RDMSS\$SET_FLAGS

The new logical name RDMSS\$SET_FLAGS accepts a string in the same format as provided to the SQL SET FLAGS statement. Abbreviations, values and negation (NO) of keywords are also supported. The equivalence string is processed after the logical name RDMSS\$DEBUG_FLAGS during attach to the database. Therefore, settings in RDMSS\$DEBUG_FLAGS will be superseded by keywords defined by this logical name. Unlike other Oracle Rdb logical names, an exception is raised if an error is found in the RDMSS\$SET_FLAGS string and the attach to the database will fail.

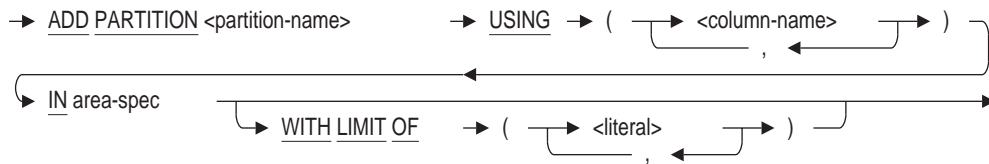
The SQL SHOW FLAGS command can be used to see which flags are set during an interactive SQL session.

7.7.15 New ADD PARTITION Clause for ALTER INDEX

The ALTER INDEX command has been enhanced with this release of Oracle Rdb7.

A new ADD PARTITION clause is now available to add a single partition within an existing HASHED index.

add-partition-clause =



Usage Notes

- The partition name is currently ignored by Oracle Rdb7. In a future release Rdb will store the name in the system table RDB\$STORAGE_MAP_AREAS so that it can be used with other partition related statements. The name will then be validated and must be unique per index.
- ADD PARTITION is currently only supported for hashed indexes. Support for sorted indexes will be provided in a future release.
- The index must have been created with a STORE clause, so that additional partitions can be added.

- There must be no active queries compiled against this table. This includes declared cursors in the current session, or other applications which have referenced the table. As with other ALTER INDEX statements exclusive access to the table is required during the current transaction.
- The USING clause must list the same column names in the same order as in the original index definition.
- If no WITH LIMIT OF clause is specified then the partition will be added at the end of the index as an OTHERWISE partition. If there is an existing OTHERWISE partition for this index then an error will be reported.
- When a new final partition or an OTHERWISE partition is successfully added, no I/O to the index is required. That is, no data in the index needs to be relocated.
- The WITH LIMIT OF clause must specify a new unique set of values for the partition. There must exist a literal value for each column listed in the USING clause.

ADD PARTITION reads the RDB\$SYSTEM_RECORD rows which are stored on each page of a mixed area and locates the hash buckets for the current index. Any hash keys which fall into the new partition will be moved (with any associated duplicates) to the new partition. Any hash keys which do not belong in the newly added area will not be moved.

Note

If this hashed index is used in a PLACEMENT VIA INDEX clause of a storage map then those placed table rows are not moved by ADD PARTITION. However, the new hashed index partition will correctly reference those rows even though they will no longer be stored adjacent to the hash bucket.

- If you attach to the database using the RESTRICTED ACCESS clause then all partitions (and system record areas) will be reserved for exclusive access. These areas will also be reserved for exclusive access if the table appears in the RESERVING clause of the current transaction (either a DECLARE TRANSACTION or SET TRANSACTION statement) with an EXCLUSIVE mode. Otherwise, the default action is to reserve the new and the following partition of the index for PROTECTED WRITE. The RDB\$SYSTEM_RECORD of the new partition is reserved for SHARED WRITE and the RDB\$SYSTEM_RECORD of the existing partition is reserved for SHARED READ mode. Using EXCLUSIVE access to the partitions will limit concurrent access to those storage areas by other users of the RDB\$SYSTEM_RECORD, for instance if there are other indices stored in that storage area. However, exclusive access has the benefit of eliminating I/O to the associated snapshot files, and reducing the virtual memory requirements of this operation. Oracle therefore recommends using EXCLUSIVE mode when possible to reduce the elapsed time of the ALTER INDEX operation. A COMMIT should be performed as soon as possible upon completion of the operation so that locks on the table are released.

If the logical name RDMSS\$CREATE_LAREA_NOLOGGING is defined then the hash buckets and duplicate nodes written to the new partition will not be journaled. However, the updates to the existing RDB\$SYSTEM_RECORD in that partition, and the deletes performed on the following partition will be journaled.

- If the INDEX_STATS flag is enabled then the ALTER INDEX command will then log messages to the RDMSS\$DEBUG_FLAGS_OUTPUT file (or SYSS\$OUTPUT if not defined) reporting the progress of the ADD PARTITION clause. INDEX_STATS can be enabled using the SET FLAGS 'INDEX_STATS' command or by defining the RDMSS\$DEBUG_FLAGS logical to "Ai" (with a lower case i). See the example below in Example 7-2.

Note

The read/write I/O statistics shown in the example are not displayed if STATISTICS COLLECTION IS DISABLED on the database, or if the logical name RDM\$BIND_STATS_ENABLED is defined to 0.

- The SHOW INDEX, or SHOW TABLE (INDEX) command will display the original source of the index definition, with the ADD PARTITION source appended. See the example below in Example 7-4. Use RMU/EXTRACT /ITEM=INDEX to see the current index definition with the additional partitions merged into the SQL CREATE INDEX syntax.

Examples

The example below use an index definition as shown in Example 7-1.

Example 7-2 shows the syntax for adding a partition before the final partition of an index.

Example 7-1 Original Index Definition

```
SQL> CREATE UNIQUE INDEX EMPLOYEES_INDEX
cont>     ON EMPLOYEES (EMPLOYEE_ID)
cont>     TYPE IS HASHED
cont>     STORE USING (EMPLOYEE_ID)
cont>     IN JOBS WITH LIMIT OF ('00999');
```

This requires that the final partition (which now follows the new partition) be scanned and matching keys moved to the new partition.

Example 7-3 shows the syntax for adding a partition after the final partition of an index. This required no I/O to the partition because there is no following partition and therefore no keys to be moved.

Example 7-4 shows the output from SHOW INDEX with the ADD PARTITION syntax appended to the original source of the index.

Example 7–2 Adding a Partition Before the Final Partition

```
SQL> SET TRANSACTION READ WRITE
cont> RESERVING EMPLOYEES for EXCLUSIVE WRITE;
SQL> ALTER INDEX EMPLOYEES_INDEX
cont> ADD PARTITION NEW_EMPS_200
cont> USING (EMPLOYEE_ID)
cont> IN EMP_INFO WITH LIMIT OF ('00200');
~Ai alter index "EMPLOYEES_INDEX" (hashed=1, ordered=0)
~Ai add partition "NEW_EMPS_200" : area "EMP_INFO"
~Ai storage area "EMP_INFO" larea=121
~Ai splitting partition #1
~Ai split complete: total 100 keys, moved 37 (dups 0)
~Ai reads: async 8 synch 16, writes: async 22 synch 0
SQL> COMMIT;
```

Example 7–3 Adding a New Final Partition

```
SQL> SET TRANSACTION READ WRITE
cont> RESERVING EMPLOYEES FOR EXCLUSIVE WRITE;
SQL> ALTER INDEX EMPLOYEES_INDEX
cont> ADD PARTITION NEW_EMPS_1400
cont> USING (EMPLOYEE_ID)
cont> IN EMPIDS_OVER WITH LIMIT OF ('01400');
~Ai alter index "EMPLOYEES_INDEX" (hashed=1, ordered=0)
~Ai add partition "NEW_EMPS_1400" : area "EMPIDS_OVER"
~Ai storage area "EMPIDS_OVER" larea=122
~Ai adding new final partition 3
SQL> COMMIT;
```

Example 7–4 Adding a Partition Before the Final Partition

```
SQL> SHOW INDEX EMPLOYEES_INDEX
Indexes on table EMPLOYEES:
EMPLOYEES_INDEX          with column EMPLOYEE_ID
  No Duplicates allowed
  Type is Hashed Scattered
  Compression is DISABLED
Store clause:           STORE using (EMPLOYEE_ID)
                        in JOBS with limit of ('00999')
                        Add Partition partition NEW_EMPS_200
                        using (EMPLOYEE_ID)
                        in EMP_INFO with limit of ('00200')
                        Add Partition partition NEW_EMPS_1400
                        using (EMPLOYEE_ID)
                        in EMPIDS_OVER with limit of ('01400')
```

7.7.16 Enhancement to the SET TRANSACTION Statement

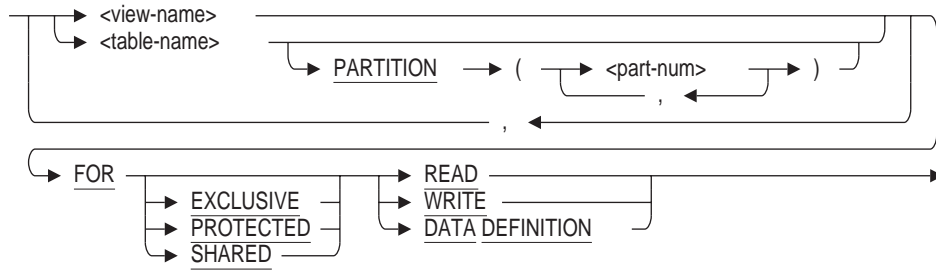
Bug 548039.

The SET TRANSACTION and DECLARE TRANSACTION statements have been enhanced for Oracle Rdb7 Release 7.0.1.3 so that selected partitions of a horizontally partitioned table can be independently reserved.

The objective is to allow concurrent partitioned operations on a single table with the highest locking modes available.

Figure 7-1 RESERVING Clause

reserving-clause =



Syntax

The changed syntax for the RESERVING clause show in Figure 7-1.

The **part-num** is the number for the partition to be reserved, or locked. Only the values in the RDB\$STORAGE_MAP_AREAS table in the RDB\$ORDINAL_POSITION column may be specified. Duplicate **part-num** values in the RESERVING clause will be ignored by SQL. Access to partitions not listed in the reserving clause will default to SHARED access.

The PARTITION clause is not permitted if a table is not mapped (has no storage map), or has a map which is vertically partitioned (uses the STORE COLUMNS clause). If an index has an identical STORE clause as the storage map then it will also be locked using the same list of partition numbers.

```
SQL> SET TRANSACTION READ WRITE
cont> RESERVING EMPLOYEES PARTITION (2) FOR EXCLUSIVE WRITE;
```

In this example just the second partition will be locked in EXCLUSIVE WRITE mode. The advantage is that this process can now insert, update or delete from this partition without writing to the snapshot file (.snp), and in general uses less resources for operations on the partition. Several processes can now concurrently update the EMPLOYEES table (providing each uses a distinct set of partitions) and use EXCLUSIVE access.

Customers should be advised that using the PARTITION clause needs careful database and application design. For instance, if the indices are partitioned using different partitioning keys, or different value ranges then cross partition updates could lead to deadlocks and other lock conflicts between the concurrent update processes.

Note

The PARTITION clause is not compatible with the DATA DEFINITION clause.

7.7.17 Computed Column Restriction Lifted for CREATE TRANSFER

Bug 572514.

Until now, SQL has imposed a restriction on the definitions of computed columns used in CREATE TRANSFER statements. The computed column definitions were not permitted to refer to domain names. If such column definitions were encountered, SQL issued the following warning message.

```
"SQL$_CMPBYWNRL, Invalid computed field <column-name> will not be transferred from relation <table-name>"
```

That column would then be removed from the list of those to be transferred.

This restriction has been removed from SQL. Removal of this restriction in SQL, however, does not completely solve the problem. If you attempt to create and execute a transfer without taking preparatory steps (see workaround farther on), execution of the transfer will fail if you are using the Replication Option for Rdb release 7.0.1 or earlier. Those versions of the Replication Option are not able to transfer the definitions of domains referenced only within computed columns.

The following example shows domain and table definitions and a CREATE TRANSFER statement which would have resulted in the SQL\$_CMPBYWNRL warning message from SQL.

```
SQL> ATTACH 'FILE DISK:[DIR]SOURCE.RDB';
SQL>
SQL> -- Create a table with two columns, one of which is computed and whose
SQL> -- definition references the name of a domain.
SQL>
SQL> CREATE DOMAIN DOM1 SMALLINT;
SQL>
SQL> CREATE TABLE TAB1 (
cont> COL1 INTEGER,
cont> COL2 COMPUTED BY
cont>     CAST(SUBSTRING(CAST(COL1 AS CHAR(4)) FROM 1 FOR 2) AS DOM1));
SQL> COMMIT;
SQL>
SQL> -- Prior to lifting the restriction in SQL, the following transfer definition
SQL> -- would have resulted in a SQL warning message: %SQL-W-CMPBYWNRL, Invalid
SQL> -- computed field COL2 will not be transferred from relation TAB1.
SQL>
SQL> CREATE TRANSFER COMPUTED_DOMAIN_REF TYPE IS EXTRACTION
cont>   MOVE TABLES TAB1
cont>   TO EXISTING FILENAME DISK:[DIR]TARGET.RDB
cont>   LOGFILE IS DISK:[DIR]COMPUTED_DOMAIN_REF.LOG;
```

To successfully perform this transfer using a version of the Replication Option for Rdb which does not transfer domains referenced within computed columns, use the following workaround. In the preceding example, using the new version of SQL, the transfer definition resulting from the CREATE TRANSFER statement would include the COL2 column to be transferred. Since the DOM1 domain is only referenced within the definition of COL2, a computed column, the Replication Option does not recreate that DOM1 definition in the target database. Therefore, prior to the first execution of the transfer, you must add the DOM1 definition to the target database yourself, using a CREATE DOMAIN statement as shown in the preceding example.

The restriction on the use of domain references within computed columns used in a CREATE TRANSFER statement has been removed from SQL in Oracle Rdb7 Release 7.0.1.3.

7.7.18 Change In Functionality for RESTRICTED ACCESS Clause

A transaction which reserves a table for EXCLUSIVE access does not also reserve the LIST area for EXCLUSIVE access. The LIST (segmented string) area is usually shared by many tables and therefore SHARED access is assumed, by default, to permit updates to the other tables.

This means that during an RMU/LOAD operation or an application update of a table reserved for EXCLUSIVE access, it may be observed that the snapshot storage area (.snp) grows. This is due to the I/O to the LIST area which is performed by default using SHARED WRITE mode.

In the original release of Oracle Rdb7, the RESTRICTED ACCESS clause on the ATTACH statement was changed so that all storage areas were accessed in EXCLUSIVE mode. This clause should be used to eliminate the snapshot I/O and related overhead when performing a lot of I/O to the LIST storage areas, such as when restructuring the database or dropping a large table containing LIST OF BYTE VARYING columns and data.

Note

RESTRICTED ACCESS is the default for SQL IMPORT, therefore, there is reduced overhead during the IMPORT of LIST data.

7.7.19 SQL Expression Support for ORDER BY and GROUP BY Clauses

Until now SQL syntax prohibited the use of expressions in either the ORDER BY or GROUP BY clauses. Now expressions are supported in both places. Note the following restrictions when using GROUP BY expressions.

- You must have a syntactically similar expression in the select list.
- The star (*) is not supported when using expressions with GROUP BY.
- GROUP BY expressions are not supported in subqueries.

The following platforms are affected by this feature:

- Interactive SQL
- SQL module language
- Precompiled SQL

The following examples show both proper and improper uses of expressions with ORDER BY and GROUP BY.

```

SQL> SELECT * FROM X ORDER BY ABS(XCOL1 - 3);
      XCOL1      XCOL2
      2          10
      1           1
      6          100
3 rows selected
SQL> SELECT (XCOL1 + 2) COL FROM X GROUP BY (XCOL1 + 2);
      COL
      3
      4
      8
3 rows selected
SQL> SELECT (2 + XCOL1) COL FROM X GROUP BY (XCOL1 + 2);
%SQL-F-NOTGROFLD, Column XCOL1 cannot be referred to in the select
list, ORDER BY, or HAVING clause because it is not in the GROUP BY clause
SQL> SELECT * FROM X GROUP BY (XCOL1 + 2);
%SQL-F-INVSELSTAR, * is not allowed in this context

```

7.7.20 [No]Commit Qualifier Added to RMU/RESTORE Command

A new qualifier, [No]Commit, has been added to the RMU/RESTORE command. This qualifier is only used when the backup file being restored is from a previous version of Oracle Rdb. Explicitly specifying the COMMIT qualifier instructs RMU to commit the converted database to the current version of Oracle Rdb before completing the restoration. In this case, the conversion is permanent and the database cannot be returned to the previous version. This is also the default behavior if the COMMIT qualifier is not used. Specifying NOCOMMIT instructs RMU not to commit the converted database. In this case, the database may later be rolled back to its original version using the RMU/CONVERT ROLLBACK command or it may be permanently committed to the current version using the RMU/CONVERT COMMIT command.

7.7.21 /WAIT Qualifier Added to RMU/OPEN Command

Previously, the RMU/OPEN command could return before a database was completely open and available. This was generally most obvious when a database was re-opened after a node failure and the database recovery processes ran for a long time.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3. A /WAIT qualifier has been added to the RMU/OPEN command. When /WAIT is specified, the RMU/OPEN command will not return until the database is open and completely recovered. At this point, the database is available for normal access. The default behavior is /NOWAIT which is the same behavior that database open has always had (the RMU/OPEN command returns before recovery completes).

7.7.22 Limit the Number and Size of AIJ Initialization I/O Buffers

When an AIJ backup operation completes, the after image journal file(s) are initialized with a pattern of -1 (hex FF) bytes. This initialization is designed to be as fast as possible and thus fully utilizes the I/O subsystem by performing many large, asynchronous I/Os at once. This speed can, however, come at the cost of a high load on I/O components during the initialization. This load could slow down other I/Os on the system.

In order to allow control over the relative I/O load that the AIJ initialization operation places on the system, two logical names have been created. These logical names should be defined in the system logical name table and are translated each time an AIJ file is initialized.

RDM\$BIND_AIJ_INITIALIZE_IO_COUNT specifies the number of asynchronous I/O operations that will be queued at once to the AIJ file. The default value if the logical name is not defined is 15, the minimum value is 1 and the maximum value is 32.

RDM\$BIND_AIJ_INITIALIZE_IO_SIZE controls the number of 512-byte disk blocks to be written per I/O. The default value, if the logical name is not defined, is 127. The minimum value is 4 and the maximum value is 127.

Reducing the value of either logical will likely increase the amount of time needed to initialize the AIJ file after a backup. However, it may also reduce load on the I/O subsystem.

7.7.23 RMU/SHOW SYSTEM and RMU/SHOW USERS Now Include Elapsed Times

The Oracle Rdb RMU/SHOW SYSTEM and RMU/SHOW USERS commands now display elapsed as well as absolute times for the time that the monitor was started and the time that databases were opened.

The following example demonstrates this output:

```
$ RMU/SHOW USERS
Oracle Rdb V7.0-13 on node HOTRDB 2-APR-1998 16:56:05.43
  - monitor started 1-APR-1998 16:51:09.37 (uptime 1 00:04:56.06)
  - monitor log filename is "DISK$:[RDM$MONITOR]RDMMON70.LOG;1"

database DISK$:[DB]MYDB.RDB;1
  - first opened 2-APR-1998 16:56:04.85 (elapsed 0 00:00:00.59)
  - 1 active database user
  - 22E07174:1 - BATCH_874 - non-utility, RDBTESTER - active user
    - image DISK$:[RDBVMS]RDBTESTER.EXE;1
```

7.7.24 New Restricted_Access Qualifier for RMU/LOAD

The RMU/LOAD command now supports the RESTRICTED_ACCESS option when attaching to an Oracle Rdb database. This option allows a single process to load data and enables some optimizations available only when RESTRICTED_ACCESS is in use.

If you are loading a table from an RMU Unload file which contains LIST OF BYTE VARYING data, the /RESTRICTED_ACCESS option will reserve the LIST areas for EXCLUSIVE access. This reduces the virtual memory used by long transactions in RMU Load, and also eliminates I/O to the snapshot files for the LIST storage areas.

The RESTRICTED_ACCESS and PARALLEL options are mutually exclusive and may not both be specified on the RMU Load command line, or within a plan file. While RMU Load is running with this option enabled, no other user may attach to the database. The default is NORESTRICTED_ACCESS.

7.7.25 New Qualifier for RMU/SHOW STATISTICS Command

The RMU/SHOW STATISTICS utility consumes approximately 13 thousand bytes of virtual memory per logical area. Also, the number of logical areas is determined by the largest logical area identifier, not the actual number of areas.

This can result in the RMU/SHOW STATISTICS utility consuming large amounts of virtual memory, even if you do not wish to review logical area statistic information.

There currently is no method available to disable the display of logical area statistic information.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.3. A new qualifier for the `RMU/SHOW STATISTICS` command, `/[NO]LOGICAL_AREA`, can be used to indicate that you do not wish to display logical area statistics information. By specifying the `/NOLOGICAL_AREA` qualifier, the virtual memory for logical area statistics information presentation will not be acquired.

Be careful when specifying the `/NOLOGICAL_AREA` qualifier that you do not specify `/NOLOG`, which will cause logical area statistic information to still be collected.

The command default is `/LOGICAL_AREA`.

There is no corresponding configuration variable. This qualifier cannot be modified at run-time.

7.7.26 `RMU/SHOW STATISTICS` "Automatic Screen Capture" Facility

The `RMU/SHOW STATISTICS` utility has been enhanced to provide an "Automatic Screen Capture" facility. This facility allows you to automatically capture images of all screens, at a specified interval. The facility is similar to using the "Options" onscreen-menu option every so often.

The "Automatic Screen Capture" facility is invoked using the "Start automatic screen capture" option of the "Tools" menu (obtained using the "!" keystroke). You will be requested to enter the interval *between* screen capture operations, expressed in seconds. The minimum interval is 30 seconds.

It takes approximately 5 to 10 seconds to capture all available screens. You will be notified when the screens are being captured by the message "**** Writing Report ****" being displayed on the status region of the current screen.

In order to guarantee consistent statistic information, statistic information updates are temporarily "paused" while the screen capture operation is occurring. Note that this "pause" also affects writing to the binary output file, as well as any log files being recorded.

The "Automatic Screen Capture" facility can be disabled using the "Stop automatic screen capture" option of the "Tools" menu.

The "Automatic Screen Capture" facility can also be invoked using the configuration variable `REPORT_INTERVAL` which specifies the number of seconds.

There is no command qualifier for this facility. Also, you cannot use the facility if the `/NOINTERACTIVE` qualifier is specified.

The "Automatic Screen Capture" facility works with binary files.

The "Automatic Screen Capture" facility is integrated with the cluster statistic collection facility. If cluster statistic collection is enabled, all supported screens will provide cluster information.

7.7.27 RMU/SHOW STATISTIC "Logical Area Overview" Screen

The RMU/SHOW STATISTIC utility has been enhanced to provide a "Logical Area Overview" screen. Located in the "Logical Area Information" sub-menu, the logical area overview screen provides a comparison of all logical areas of a particular type.

The following is an example of the "Logical Area Overview" screen:

```
Node: ALPHA3 (1/1/16)   Oracle Rdb X7.0-00 Perf. Monitor 18-MAR-1998 14:20:54.98
Rate: 1.00 Second      Logical Area Overview (Tables)      Elapsed: 03:28:56.70
Page: 1 of 1          KODH$:[R_ANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1   Mode: Online
```

```
-----
Logical.Area.Name... record fetch record store record erase      discarded CurTot
RDB$RELATIONS.RDB$SY          29          0          0          0
RDB$FIELD_VERSIONS.R        217          0          0          0
RDB$INDICES.RDB$SYST         35          0          0          0
RDB$INDEX_SEGMENTS.R        35          0          0          0
RDB$FIELDS.RDB$SYSTE        12          0          0          0
RDB$RELATION_FIELDS.        12          0          0          0
RDB$DATABASE.RDB$SYS         1          0          0          0
RDB$VIEW_RELATIONS.R         0          0          0          0
RDB$CONSTRAINT_RELAT         6          0          0          0
RDB$CONSTRAINTS.RDB$         6          0          0          0
RDB$STORAGE_MAPS.RDB         9          0          0          0
RDB$STORAGE_MAP_AREA        17          0          0          0
RDB$INTERRELATIONS.R         0          0          0          0
RDB$COLLATIONS.RDB$S         0          0          0          0
RDB$TRIGGERS.RDB$SYS         1          0          0          0
RDB$RELATION_CONSTRA         0          0          0          0
RDB$RELATION_CONSTRA         0          0          0          0
RDB$PRIVILEGES.RDB$S         0          0          0          0
RDB$MODULES.RDB$SYST         0          0          0          0
RDB$ROUTINES.RDB$SYS         0          0          0          0
RDB$PARAMETERS.RDB$S         0          0          0          0
RDB$QUERY_OUTLINES.R         0          0          0          0
RDB$WORKLOAD.RDB$SYS         37          0          0          0
CANDIDATES.RDB$SYSTE         0          0          0          0
COLLEGES.EMP_INFO            0          0          0          0
DEGREES.EMP_INFO             495          0          165          0
DEPARTMENTS.DEPARTME        2626          0          0          0
EMPLOYEES.EMPIDS_LOW         148          0          37          0
EMPLOYEES.EMPIDS_MID         228          0          57          0
EMPLOYEES.EMPIDS_OVE         24          0          6          0
JOBS.JOBS                     0          0          0          0
JOB_HISTORY.EMPIDS_L         306          0          102          0
JOB_HISTORY.EMPIDS_M         450          0          150          0
JOB_HISTORY.EMPIDS_O         66          0          22          0
RESUMES.EMP_INFO             58600          0          0          0
SALARY_HISTORY.SALAR         2187          0          729          0
WORK_STATUS.EMP_INFO         0          0          0          0
-----
```

```
Config Exit Help Menu >next_page <prev_page Options Pause Reset Set_rate Write
```

The "Logical Area Overview" screen displays the following information:

- **Logical Area Name.** This column displays the name of the logical area, followed by a period ("."), followed by the name of the physical area (storage area) in which the logical area partition resides.

A maximum of 20 characters is displayed, which typically results in the storage area name being truncated.

For performance reasons, the logical area names are *not* sorted in any particular order.

- **Statistic Field #1.** This column displays a user-selectable statistic field appropriate for the logical area type.
The default statistic field is the following:
 - Table - record fetch.
 - B-tree Index - leaf fetches.
 - Hash Index - hash index fetched.
 - Blob - blob fetched.
- **Statistic Field #2.** This column displays a user-selectable statistic field appropriate for the logical area type.
The default statistic field is the following:
 - Table - record stored.
 - B-tree Index - leaf insertion.
 - Hash Index - hash insertion.
 - Blob - blob stored.
- **Statistic Field #3.** This column displays a user-selectable statistic field appropriate for the logical area type.
The default statistic field is the following:
 - Table - record erased.
 - B-tree Index - leaf removal.
 - Hash Index - hash deletion.
 - Blob - blob erased.
- **Statistic Field #4.** This column displays a user-selectable statistic field appropriate for the logical area type.
The default statistic field is the following:
 - Table - pages discarded.
 - B-tree Index - pages discarded.
 - Hash Index - pages discarded.
 - Blob - pages discarded.
- **Statistic Type.** This column identifies the “type” of statistic information being displayed. The following types are available:
 - **CurTot** - Current total.
 - **CurRate** - Current rate.
 - **MaxRate** - Maximum rate.
 - **AvgRate** - Average rate.
 - **PerTrans** - Per-transaction rate.

Selecting the “Logical Area Information” menu will now display two options: “Logical Area Overview (type)” and the previously existing “Logical Area Statistics”.

The “system” logical areas can be filtered from the “Logical Area Overview” screen by selecting the “Display application logical areas” option of the “Tools” menu (obtained using the “!” shortcut). System logical areas can be included on the screen by selecting the “Display all logical areas” option of the “Tools” menu.

The “Logical Area Overview” screen statistic type can be specified using the configuration variable **LOGICAL_OVERVIEW_STAT** with the following keywords **CUR_TOTAL**, **CUR_RATE**, **MAX_RATE**, **AVG_RATE** and **PER_TRANS**.

The “Logical Area Overview” screen logical area type can be specified using the configuration variable **LOGICAL_OVERVIEW_TYPE** with the following keywords **TABLE**, **BTREE**, **HASH** and **BLOB**.

The “Logical Area Overview” screen can be configured to display application logical areas only (no “system” logical areas) using the configuration variable **SYSTEM_LOGICAL_AREAS** with the keyword **FALSE**. Specifying the configuration variable with the keyword **TRUE**, the default, will display all logical areas, including “system” logical areas.

The “Logical Area Overview” screen information is not saved in the binary output and, therefore, the screen is not available during binary file replay.

The “Logical Area Overview” screen is not available if the **/NOLOGICAL_AREA** qualifier is specified.

The “Logical Area Overview” screen participates in the “Cluster Statistic Collection” facility.

The following screen configuration options are available using the “Config” onscreen-menu option:

- **Modify column #1.** This option allows you to choose a different statistic field for column number 1.
- **Modify column #2.** This option allows you to choose a different statistic field for column number 2.
- **Modify column #3.** This option allows you to choose a different statistic field for column number 3.
- **Modify column #4.** This option allows you to choose a different statistic field for column number 4.
- **Change logical area type.** This option allows you to choose a different logical area type to be displayed on the screen. Selecting a new logical area type will reset the statistic fields to the default fields for that logical area type.

Logical area types are: table, B-tree index, hash index and blob.

- **Change statistic type.** This option allows you to choose a different statistic type to be displayed on the screen. The selected statistic type applies to all statistic fields on the screen.

Statistic types are: current total, current rate, maximum rate, average rate and per-transaction rate.

When selecting statistic fields for the various columns, no validation is performed to eliminate duplicate selections. This means you can display the same statistic field in one or more columns at the same time, if you so desire.

The following is an example of the "Logical Area Overview" screen for B-tree index logical areas:

```
Node: ALPHA3 (1/1/16) Oracle Rdb X7.0-00 Perf. Monitor 18-MAR-1998 15:10:40.79
Rate: 1.00 Second Logical Area Overview (Btree Indexes) Elapsed: 04:18:42.51
Page: 1 of 1 KODH$: [R_ANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1 Mode: Online
```

```
-----
Logical.Area.Name... leaf fetches leaf inserti leaf removal discarded CurTot
COLL_COLLEGE_CODE.RD          0          0          0          0
DEG_EMP_ID.RDB$SYSTE        103          0          5          0
DEPARTMENTS_INDEX.DE        100          0          0          0
EMP_EMPLOYEE_ID.RDB$         5          0          3          0
JH_EMPLOYEE_ID.RDB$$         1          0          3          0
SH_EMPLOYEE_ID.RDB$$        103          0          3          0
-----
```

```
-----
Config Exit Help Menu >next_page <prev_page Options Pause Reset Set_rate Write
```

The following is an example of the "Logical Area Overview" screen for hash index logical areas:

```
Node: ALPHA3 (1/1/16) Oracle Rdb X7.0-00 Perf. Monitor 18-MAR-1998 15:11:09.68
Rate: 1.00 Second Logical Area Overview (Hash Indexes) Elapsed: 04:19:11.40
Page: 1 of 1 KODH$: [R_ANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1 Mode: Online
```

```
-----
Logical.Area.Name... hash index f hash inserti hash deletio discarded CurTot
EMPLOYEES_HASH.EMPID         37          0          37          0
EMPLOYEES_HASH.EMPID         57          0          57          0
EMPLOYEES_HASH.EMPID          6          0           6          0
JOB_HISTORY_HASH.EMP         235         0          37          0
JOB_HISTORY_HASH.EMP         343         0          57          0
JOB_HISTORY_HASH.EMP          50          0           6          0
-----
```

```
-----
Config Exit Help Menu >next_page <prev_page Options Pause Reset Set_rate Write
```

The following is an example of the "Logical Area Overview" screen for blob logical areas:

```
Node: ALPHA3 (1/1/16) Oracle Rdb X7.0-00 Perf. Monitor 18-MAR-1998 15:11:38.15
Rate: 1.00 Second Logical Area Overview (Blobs) Elapsed: 04:19:39.87
Page: 1 of 1 KODH$: [R_ANDERSON.WORK.STATS]MF_PERSONNEL.RDB;1 Mode: Online
```

```
-----
Logical.Area.Name... blob fetched blob stored blob erased discarded CurTot
RDB$SEGMENTED_STRING         73          0          0          0
-----
```

```
-----
Config Exit Help Menu >next_page <prev_page Options Pause Reset Set_rate Write
```

The "Logical Area Information" screen can also be sorted alphabetically and the user can "zoom in" on any displayed logical area to display that area's actual statistic information.

7.7.28 RMU/SHOW STATISTICS "Summary Tx Statistics" Screen

The **RMU/SHOW STATISTICS** utility has been enhanced to provide a "Summary Tx Statistics" screen. This screen summarizes database transaction activity and indicates transaction and verb execution rates.

The information displayed on the screen is a summation of the information displayed on a per-storage-area basis in the "IO Statistics" screens. This screen resides in the "Main" menu.

The following is an example of this screen:

```
Node: ALPHA3 (1/3/24)   Oracle Rdb X7.0-00 Perf. Monitor 10-JUL-1998 10:09:22.31
Rate: 1.00 Second      Summary Tx Statistics           Elapsed: 01:13:09.40
Page: 1 of 1          KODA_TEST:[R_ANDERSON.TCS_MASTER]TCS.RDB;2   Mode: Online
```

```
-----
statistic..... rate.per.second..... total..... average.....
name..... max..... cur..... avg..... count..... per.trans....
transactions          4          0          0.4          2065          1.0
  committed           4          0          0.4          2065          1.0
  rolled back         0          0          0.0           0          0.0
  duration x100       0          0          0.0           0          0.0
  prepared            0          0          0.0           0          0.0
verb successes        455          6          27.1        119362          57.8
verb failures          0          0          0.0           0          0.0
  duration x100       0          0          0.0           0          0.0
checkpoints           1          0          0.0           42          0.0
  duration x100      133552          0          61.4        269839        130.6
RUJ file reads         0          0          0.0           0          0.0
  file writes         4          1          0.5          2516          1.1
  file extend         1          0          0.0           313          0.1
-----
```

Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot Yank !

Table 7-4 Screen Fields

Field	Description
transactions	The number of completed database transactions. This is the count of the COMMIT and ROLLBACK statements that have executed.
committed	The number of transactions that successfully updated the database.
rolled back	The number of transactions that were aborted and were not applied to the database.
duration x100	The duration of a transaction rollback operation, expressed in hundredths of a second displayed as a whole number. For example, the value "500" is "5" seconds.
prepared	The number of distributed transactions that have successfully "prepared" themselves for subsequent transaction commit.
verb successes	The number of completed verbs that returned a success status code. A verb is an atomic SQL statement or action. For example, a record insert and a record deletion are verbs. Also, within a compound statement each individual statement is atomic and Oracle Rdb performs a verb-success operation after processing each one. To avoid this overhead, you can use the SQL BEGIN ATOMIC statement to treat the entire block as a single verb.
verb failures	Give the number of completed verbs that returned an error status code. Errors include end-of-collection and deadlocks, as well as other exception conditions.

(continued on next page)

Table 7–4 (Cont.) Screen Fields

Field	Description
duration x100	Identifies the duration of a verb failure rollback operation, expressed in hundredths of a second displayed as a whole number. For example, the value “500” is “5” seconds.
checkpoints	Identifies the number of checkpoints performed by users. This field does not include the initial checkpoint when the user first attaches to the database.
duration x100	Displays the checkpoint duration, expressed in hundredths of a second displayed as a whole number. For example, the value “500” is “5” seconds.
RUJ file reads	Displays the total number of read I/O operations performed on the RUJ journal during the transaction undo phase. The RUJ file is never written by the database recovery (DBR) process. This field includes both synchronous and asynchronous I/O read requests.
file writes	Displays the total number of write I/O operations performed on the RUJ journal during the transaction phase. This field includes both synchronous and asynchronous I/O read requests.
file extends	Identifies the number of times an RUJ file has been extended.

7.7.29 RMU/SHOW STATISTICS "Recovery Information" Screen

This screen provides run-time standby database recovery information. It is important for analyzing network bandwidth utilization and standby database resource allocation effectiveness

This screen is only available on the standby database while Hot Standby is active. It resides in the “Hot Standby Information” menu.

The following is an example of the “Recovery Information” screen:

```

-----
statistic..... rate.per.second..... total..... average.....
name..... max..... cur..... avg..... count..... per.trans....
transactions          33          0          0.4          1979          1.0
  commit              33          0          0.4          1979          1.0
  rollback             0          0          0.0           0          0.0
  prepared             0          0          0.0           0          0.0
Area ready            0          0          0.0           12          0.0
AIJ records          3030          0          86.6         399769         202.0
  erase mixed           0          0          0.0           0          0.0
  erase uniform          0          0          0.0           0          0.0
  modify mixed         1876          0          16.3         75399          38.0
  modify uniform       3030          0          70.3        324370         163.9
SPAM updated          806          0          13.8         63716          32.1
-----

```

Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot Yank !

Table 7-5 Screen Fields

Field	Description
transactions	Gives the number of completed database transactions. This is the count of the COMMIT and ROLLBACK statements that have executed.
commit	Displays the number of transactions that have been committed to the standby database.
rollback	Displays the number of transactions that have been rolled back prior to being applied to the standby database.
prepared	Displays the number of distributed transactions that have been successfully "prepared" in anticipation of eventually being committed to the standby database.
Area ready	Displays the number of physical storage areas that have been "readied" during the recovery operation.
AIJ records	Displays the number of AIJ records applied.
erase mixed	Displays the number of "erase record" operations performed on a mixed-format storage area.
erase uniform	Displays the number of "erase record" operations performed on a uniform-format storage area.
modify mixed	Displays the number of "modify record" operations performed on a mixed-format storage area.
modify uniform	Displays the number of "modify record" operations performed on a uniform-format storage area.
SPAM updated	Displays the number of SPAM page modifications that occurred as a result of the AIJ journal record. SPAM pages are typically modified due to a live data page changing its threshold information.

7.8 Enhancements Provided In Oracle Rdb7 Release 7.0.1.2

7.8.1 Monitor Process Uses Less ENQLM

The Oracle Rdb7 monitor process holds null mode locks on a number of database resources in order to keep the lock value blocks valid even when there are no users attached to a database. For systems that have a large number of databases or databases with a large number of storage areas, the monitor process can consume a great number of locks, sometimes exceeding its lock quota (ENQLM) even at the OpenVMS maximum value of 32767 locks.

The impact of this situation has been reduced in Oracle Rdb7 Release 7.0.1.2. By using the LCK\$M_NOQUOTA flag when taking out many of these locks (in particular, the database FILID, SAC, RCACHE, TSNBLK and SEQBLK locks), the monitor process uses less of its ENQLM. The total number of locks and system resources consumed remains the same but the monitor process's ENQLM is not deducted for these locks.

7.8.2 RDMS\$TTB_HASH_SIZE Logical Name

Temporary table users should be aware of a new logical name being added to Oracle Rdb7. The temporary table code uses a hash table that is sized according to the setting of the RDMS\$TTB_HASH_SIZE logical name. If the logical has not been defined, a default value of 1249 will be used.

If expected usage is such that temporary tables will be large (10k or more rows), this logical should be used to adjust the hash table size used to avoid long hash chains. The setting of this logical should be on the order of roughly 1/4 of the expected maximum number of rows per temporary table. So, if its likely that a temporary table will be populated with 100,000 rows, then define this logical to be 25,000. But if there are memory constraints, it is advisable that the logical be defined no higher than this value.

7.8.3 New SQLSTATE Value

If a SQL statement expects a value from a function which does not return a value, the SQLSTATE value will be set to '2F001' to reflect the error state.

This new error code is shown in the following example.


```

SQL> CREATE DATABASE FILE TEST2;
SQL>     SET DIALECT 'SQL92';
SQL>
SQL>     CREATE MODULE RETURN_M
cont>         LANGUAGE sql
cont>
cont>         FUNCTION RETURN_F (:A INTEGER)
cont>             RETURNS INTEGER;
cont>             BEGIN
cont>                 IF :A IS NOT NULL THEN
cont>                     RETURN - :A;
cont>                 END IF;
cont>             END;
cont>     END MODULE;
SQL>
SQL>     SELECT RETURN_F (NULL) FROM RDB$DATABASE;
%RDB-F-NORESULT, stored function "RETURN_F" returned no result
-RDB-F-ON_DB, on database SQL_USER4:[USER.DB]TEST2.RDB;
SQL> SHOW SQLCA
SQLCA:
      SQLCAID:      SQLCA          SQLCABC:      128
      SQLCODE:      -1043
      SQLERRD:      [0]: 0
                   [1]: 0
                   [2]: 0
                   [3]: 0
                   [4]: 0
                   [5]: 0
      SQLWARN0:      0          SQLWARN1:      0          SQLWARN2:      0
      SQLWARN3:      0          SQLWARN4:      0          SQLWARN5:      0
      SQLWARN6:      0          SQLWARN7:      0
      SQLSTATE:      2F001
SQL> ROLLBACK;
SQL> DROP DATABASE FILE TEST2;

```

7.8.4 Planned Change in Behavior for the UNIQUE Predicate

The next major release of Oracle Rdb will change the behavior of the UNIQUE predicate. Up to the Oracle Rdb7 release there was no semantic difference between the undocumented UNIQUE predicate and the documented SINGLE predicate. This will change with the release of Oracle Rdb Release 7.1.

The UNIQUE predicate in Oracle Rdb was originally implemented for compatibility with the RDO interface and as such required that exactly one row matched, this included a single column value set to NULL. However, these semantics do not match the current SQL database language standard SQL92 for the UNIQUE predicate. Therefore, the syntax was deprecated and replaced with SINGLE.

When SINGLE is used, then a single matching row is required for uniqueness. Zero, or more than one row will be considered non-unique. The syntax and semantics of SINGLE will not be changed. If applications currently use the UNIQUE predicate, but require these semantics, then applications must be changed to use the SINGLE predicate.

The syntax for UNIQUE has been deprecated for many versions in preparation for this change in behavior in compliance with the current SQL database language standard. An example of the deprecated message, follows:

```

SQL> SELECT EMPLOYEE_ID
cont> FROM EMPLOYEES
cont> WHERE UNIQUE (SELECT EMPLOYEE_ID FROM JOB_HISTORY);
%SQL-I-DEPR_FEATURE, Deprecated Feature: UNIQUE is replaced by SINGLE

```

In Oracle Rdb Release 7.1, the UNIQUE predicate will be documented and the deprecated message will no longer be used. The changed semantics may cause additional rows to be returned from queries, because now rows with column values set to NULL will always be considered UNIQUE.

Note

This topic is an announcement of a future new feature for Oracle Rdb Release 7.1. Use the information contained in it for planning purposes only with Oracle Rdb7 Release 7.0.1.2.

7.8.5 UNION ALL and Derived Tables Allow up to 2000 Value Expressions

The DISTINCT, ORDER BY, GROUP BY, and UNION clauses are restricted to 255 value expressions in all releases of Rdb7 due to restrictions in processing DISTINCT and ORDER BY clauses.

Unlike UNION, the UNION ALL clause does not perform an implicit DISTINCT operation and so need not be restricted in the same way as the UNION clause. Therefore, in Oracle Rdb7 Release 7.0.1.2 the UNION ALL clause now allows up to 2000 value expressions.

The restriction of 255 column names for a derived table has also been lifted so that now up to 2000 columns can be visible through a derived table expression.

If older versions of Oracle Rdb7 are remotely accessed, then the previous limits will still be imposed.

7.8.6 RMU/DUMP/AFTER Command /START and /END Qualifiers Improved

The /START and /END qualifiers for the RMU/DUMP/AFTER_JOURNAL command were difficult to use because users seldom know, nor can they determine, the AIJ record number in advance of using the command.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The RMU/DUMP/AFTER_JOURNAL command has been enhanced to provide more advanced selection criteria. Three new optional qualifiers, /FIRST=select_list, /LAST=select_list, and /ONLY=select_list have been added.

The select_list clause of these qualifiers consists of a list of one or more of the following keywords:

- TSN=tsn
Specifies the first, last, or specific TSN in the AIJ journal, using the standard “[n:]m” TSN format.
- TID=tid
Specifies the first, last or specific TID in the AIJ journal.
- RECORD=record
Specifies the first or last record in the AIJ journal. This is the same as the existing /START and /END qualifiers, which are still supported, but obsolete. This keyword cannot be used with the /ONLY qualifier.
- BLOCK=block#
Specifies the first or last block in the AIJ journal. This keyword cannot be used with the /ONLY qualifier.
- TIME=date_time

Specifies the first or last date/time in the AIJ journal, using the standard date/time format. This keyword cannot be used with the /ONLY qualifier.

The /FIRST, /LAST, and /ONLY qualifiers are optional. You may specify any or none of them.

The keywords specified for the /FIRST qualifier can differ from the keywords specified for the other qualifiers.

For example, to start the dump from the fifth block of the AIJ journal, you would use the following command:

```
RMU/DUMP/AFTER_JOURNAL /FIRST=(BLOCK=5) MF_PERSONNEL.AIJ
```

To start the dump from block 100 or TSN 52, whichever occurs first, you would use the following command:

```
RMU/DUMP/AFTER_JOURNAL /FIRST=(BLOCK=100,TSN=0:52) MF_PERSONNEL.AIJ
```

When multiple keywords are specified for a qualifier, the first condition being encountered activates the qualifier. In the above example, the dump will start when *either* block 100 or TSN 52 is encountered.

Note

Be careful when searching for TSNs or TIDs, as they are not ordered in the AIJ journal. For example, if you want to search for a specific TSN then use the /ONLY qualifier, not the /FIRST and /LAST qualifiers.

For example, assume the AIJ journal contains records for TSN 150, 170 and 160 (in that order). If you specify the /FIRST=TSN=160 and /LAST=TSN=160 qualifiers, nothing will be dumped because the TSN 170 will match the /LAST=TSN=160 criteria.

7.8.7 RMU/SHOW STATISTICS "Stall Message Logfile" Option Real Time Lock Information

The RMU/SHOW STATISTICS utility "Stall Message Logging" facility now shows expanded information for DBAs. It now displays real-time lock information when the displayed stall is on a lock or locked object. Both the waiting process and the blocking process are displayed. The RMU/SHOW STATISTICS "Stall Message Logging" facility now provides real-time lock information when the displayed stall is on a lock or locked object.

The following example shows the new output of a sample stall messages logfile.

```

Oracle Rdb X7.0-00 Performance Monitor Stall Log
Database USR1$:[WORK.STATS]MF_PERSONNEL.RDB;1
Stall Log created 30-SEP-1997 07:01:15.64
2AA8587A:1 08:11:54.27 reading pages 11:7534416 to 3:78
2AAA9E7B:1 08:11:54.31 waiting for async-write of pages 5:1412 to 5:1412
2AA810A7:1 08:11:54.29 waiting for page 5:876 (PW)
      State... Process.ID Process.name... Lock.ID. Rq Gr Queue page 876
      Blocker: 2AAA9E7B RICK10..... 7D00562C PR PR Grant
      Waiting: 2AA810A7 RICK13..... 71002E7D PW NL Wait
2AA8587A:1 08:11:55.34 waiting for page 5:1303 (PW)
      State... Process.ID Process.name... Lock.ID. Rq Gr Queue page 1303
      Owner: 2AA7D07C RICK11..... 31007E07 PR CR Grant
      Blocker: 2AAA9E7B RICK10..... 5A00BA0E PR PR Grant
      Waiting: 2AA8587A RICK9..... 5C005FAD PW CR Cnvrt
2AAA9E7B:1 08:11:55.37 locking page 5:565
2AA810A7:1 08:11:55.38 reading pages 5:912 to 5:914
2AAA9E7B:1 08:11:57.77 waiting for page 5:1303 (PW)
      State... Process.ID Process.name... Lock.ID. Rq Gr Queue page 1303
      Owner: 2AA810A7 RICK13..... 0C007752 PR CR Grant
      Blocker: 2AA8587A RICK9..... 2D001C3D PR PR Grant
      Waiting: 2AAA9E7B RICK10..... 47003DC3 PW CR Cnvrt
2AA7D07C:1 08:11:57.78 reading pages 5:1337 to 5:1339
2AA8587A:1 08:11:57.86 reading pages 5:330 to 5:332
2AA7D07C:1 08:11:57.86 waiting for page 5:1413 (PR)
      State... Process.ID Process.name... Lock.ID. Rq Gr Queue page 1413
      Blocker: 2AAA9E7B RICK10..... 6A002CBB PW PW Grant
      Owner: 2AA8587A RICK9..... 6F008623 PR CR Grant
      Waiting: 2AA7D07C RICK11..... 1F007B4D PR NL Wait
      .
      .
      .

```

7.8.8 RMU/SHOW STATISTICS Utility "Stall Messages Log" Displays Stall Duration Information

The RMU/SHOW STATISTICS utility “Stall Messages Logging” facility has been enhanced to provide the information necessary to determine stall duration.

First, the current time has been added to each stall message. This allows you to determine the stall duration at that point-in-time, because the stall start-time is also displayed.

Secondly, a new qualifier has been added: /OPTION=VERBOSE. This qualifier causes the stall message logging facility to report a stall message at each interval, even if it has been previously reported.

Note

Use of the /OPTION=VERBOSE qualifier could result in an enormous stall messages logfile. Ensure that adequate disk space exists for the logfile when using this qualifier.

The stall messages logging “verbose” option can be enabled and disabled at runtime, using the “Tools” menu, by pressing the “!” key.

The verbose option can also be specified in the configuration file, using the STALL_LOG_VERBOSE variable. Valid keywords are ENABLED or DISABLED.

The following example shows a stall messages logfile, in “verbose” mode, for a database where four processes are all stalled on the same lock. Note that the first stall message already indicates a 25-minute stall.

```
Oracle Rdb X7.0-00 Performance Monitor Stall Log
Database USR1$:[WORK.STATS]MF_PERSONNEL.RDB;1
Stall Log created 2-OCT-1997 09:26:15.19
09:26:15.19 2AA8C6D7:1 09:01:01.29 waiting for logical area 58 (CW)
State... Process.ID Process.name... Lock.ID. Rq Gr Queue logical
area 58
Blocker: 2AA00443 RICK2..... 7300845F PW PW Grant
Waiting: 2AA8C6D7 RICK6..... 4E008184 CW NL Cnvrt
Waiting: 2AA912D8 RICK7..... 5D0034F2 CW NL Cnvrt
Waiting: 2AA3BADC RICK8..... 0700115F CW NL Cnvrt
Waiting: 2AA43ADE RICK9..... 4700AE41 CW NL Cnvrt
09:26:15.19 2AA3BADC:1 09:01:01.37 waiting for logical area 58 (CW)
State... Process.ID Process.name... Lock.ID. Rq Gr Queue logical
area 58
Blocker: 2AA00443 RICK2..... 7300845F PW PW Grant
Waiting: 2AA8C6D7 RICK6..... 4E008184 CW NL Cnvrt
Waiting: 2AA912D8 RICK7..... 5D0034F2 CW NL Cnvrt
Waiting: 2AA3BADC RICK8..... 0700115F CW NL Cnvrt
Waiting: 2AA43ADE RICK9..... 4700AE41 CW NL Cnvrt
09:26:15.19 2AA912D8:1 09:01:01.32 waiting for logical area 58 (CW)
State... Process.ID Process.name... Lock.ID. Rq Gr Queue logical
area 58
Blocker: 2AA00443 RICK2..... 7300845F PW PW Grant
Waiting: 2AA8C6D7 RICK6..... 4E008184 CW NL Cnvrt
Waiting: 2AA912D8 RICK7..... 5D0034F2 CW NL Cnvrt
Waiting: 2AA3BADC RICK8..... 0700115F CW NL Cnvrt
Waiting: 2AA43ADE RICK9..... 4700AE41 CW NL Cnvrt
.
.
.
```

The lock information is only displayed once per stall, even in verbose mode, to minimize the the output file size.

7.8.9 RMU/SHOW STATISTICS "User-Defined Events" Enhancements

The following enhancements have been made to the RMU/SHOW STATISTICS utility “User-Defined Events” facility and the “Configuration File” facility in general:

- Long configuration file lines can be continued on the next line by terminating the line with a back-slash (“\”). Lines can be continued up to 2048 characters, even within quoted values; for example:

```
EVENT_DESCRIPTION="ENABLE 'pages checked' MAX_CUR_TOTAL \
INITIAL 7 \
EVERY 11 \
LIMIT 100 \
INVOKE DB_ALERT";
```

This enhancement is not limited to just the EVENT_DESCRIPTION variable; it can be used for any configuration variable. Also, comments can be embedded in continued lines if they start at the beginning of the next line. For example, consider the following two event descriptions:

```

EVENT_DESCRIPTION="ENABLE ' (Asynch. reads)' MAX_CUR_TOTAL \
! this will work as expected
AREA EMPIDS_OVER \
INITIAL 6 EVERY 10 LIMIT 100 INVOKE DB_ALERT";
EVENT_DESCRIPTION="ENABLE ' (Asynch. reads)' MAX_CUR_TOTAL \
AREA EMPIDS_OVER ! this will NOT work as expected \
INITIAL 6 EVERY 10 LIMIT 100 INVOKE DB_ALERT";

```

Note that the comment in the second event description takes precedence over the line continuation character.

- In the EVENT_DESCRIPTION variable value, the underscore character (“_”) or dash character (“-”) can be used in place of spaces in statistics names that have leading spaces. For example, the statistics field name “ file extend” can also be specified as “_ _ file_extend” or “- - file-extend”. This is useful for improving the readability of difficult statistics field names.
- The keyword “AREA” has been added to the user-defined event attribute list. The keyword “AREA” allows you to specify the name of a storage area. When this keyword is specified, the statistics field selected must be from the “IO Statistics (by file)” or “Locking Statistics (by file)” screens.

The **AREA** attribute is available when using the /NOINTERACTIVE qualifier, or when using the “INTERACTIVE” configuration variable set to **FALSE**.

- The keyword “LAREA” has been added to the user-defined event attribute list. The keyword “LAREA” allows you to specify the name of a logical area, which can be either a table, B-tree index, hash index or blob. When this keyword is specified, the statistics field selected must be from the “Logical Area” screens.

If the logical area is partitioned across multiple storage areas, the keyword “AREA” can be used to identify a specific partition to define the event against.

The LAREA attribute is available when using the /NOINTERACTIVE qualifier, or when using the “INTERACTIVE” configuration variable set to **FALSE**.

The following table explains the semantics of specifying the AREA and LAREA keywords:

AREA	LAREA	Description
No	No	Regular statistic field used
Yes	No	Storage Area statistic field used
No	Yes	Logical Area statistic field used - all partitions
Yes	Yes	Logical Area statistic field used - single partition

This example demonstrates how to define an event on a storage area statistic:

```

EVENT_DESCRIPTION="ENABLE ' (Asynch. reads)' MAX_CUR_TOTAL \
AREA EMPIDS_OVER \
INITIAL 6 EVERY 10 LIMIT 100 INVOKE DB_ALERT";

```

This example demonstrates how to define an event on a table. Note that this event is defined across all partitions of the table.

```

EVENT_DESCRIPTION="ENABLE 'pages checked' MAX_CUR_TOTAL \
LAREA EMPLOYEES \
INITIAL 1 EVERY 1 LIMIT 100 INVOKE DB_ALERT";

```

This example demonstrates how to define an event on a single-partition of a partitioned table.

```
EVENT_DESCRIPTION="ENABLE 'pages checked' MAX_CUR_TOTAL \
LAREA EMPLOYEES AREA EMPIDS_LOW \
INITIAL 3 EVERY 7 LIMIT 100 INVOKE DB_ALERT";
```

The "Statistics Event Information" screen has been enhanced to identify the physical area ID and logical area ID for each event. The area identifiers are displayed when using "Full" display-mode. For example, using the above examples, the screen would appear as follows:

```
Node: ALPH (1/1/2) Oracle Rdb X7.0-00 Perf. Monitor 21-OCT-1997 13:41:50.06
Rate: 1.00 Second Statistics Event Information Elapsed: 02:30:21.57
Page: 1 of 1 DISK$:[WORK]MF_PERSONNEL.RDB;1 Mode: Online
```

```
-----
Statistic..... Event..... State... Threshold Every Current Cnt
Program/Operator.Notification..... Parea Larea Rem Limit
synch data reads MAX_CUR_TOTAL enabled 228.0 11 228.0 1
DB_ALERT (@SYS$DISK:[ ]EVENT.COM) 0 0 0 100
locks requested MAX_CUR_TOTAL enabled 406.0 10 406.0 1
DB_ALERT (@SYS$DISK:[ ]EVENT.COM) 5 0 0 100
pages checked MAX_CUR_TOTAL enabled 3.0 7 0.0 0
DB_ALERT (@SYS$DISK:[ ]EVENT.COM) 3 56 0 100
pages checked MAX_CUR_TOTAL enabled 4.0 8 0.0 0
DB_ALERT (@SYS$DISK:[ ]EVENT.COM) 4 57 0 100
pages checked MAX_CUR_TOTAL enabled 10717.0 9 10717.0 1
DB_ALERT (@SYS$DISK:[ ]EVENT.COM) 5 58 0 100
pages checked MAX_CUR_TOTAL enabled 10717.0 1 10717.0 1
DB_ALERT (@SYS$DISK:[ ]EVENT.COM) 0 2 0 100
-----
```

```
Brief Config Exit Help Menu >next_page <prev_page Options Pause Set_rate Write
```

If an event on a storage area or logical area is encountered, the storage area name, the logical area name, or both are passed as the eighth parameter to the invocation program. For example, the DB_INVOKE program defined above causes the DCL script "EVENT.COM" to be executed. This DCL script simply appends the raised event to a log file; for example:

```
$ SET NOON
$ OPEN/APPEND/SHARE=READ EVENT_LOG SYS$DISK:[ ]EVENT.LOG
$ WRITE EVENT_LOG "'P1' 'P2' 'P3' 'P4' 'P5' 'P6' COUNT IS 'P7' 'P8'"
$ CLOSE EVENT_LOG
$ EXIT
```

Note that the "P8" parameter is either null ("") or contains the name of the target storage area or logical area. The following is an example of the logfile output:

```
20-OCT-1997 14:02:21.41 pages checked MAX_CUR_TOTAL 6.0 above 4.0 count is 1
EMPIDS_MID.EMPLOYEES
20-OCT-1997 14:02:22.16 pages checked MAX_CUR_TOTAL 32820.0 above 5.0 count is 1
EMPIDS_OVER.EMPLOYEES
```

Note that when both the storage area and logical area names are specified, they are separated by a period (".").

7.8.10 Added Detail to RMU/SHOW STATISTICS "SPAM Fetches" Screen

The RMU/SHOW STATISTICS utility "SPAM Fetches" screen did not display the reason why a SPAM page was fetched. This information is vital in determining when excessive SPAM fetches are occurring.

The following example shows a sample "PIO Statistics-SPAM Fetches" screen display. It is extremely difficult to determine what caused the 17,821 SPAM fetches as well as the 2,250 SPAM updates.

```
Node: ALPH (1/1/16) Oracle Rdb X7.0-00 Perf. Monitor 5-JAN-1998 11:27:51.18
Rate: 1.00 Second PIO Statistics--SPAM Fetches Elapsed: 00:30:17.60
Page: 1 of 1 DISK1$:[WORK]MF_PERSONNEL.RDB;1 Mode: Online
```

```
-----
```

statistic..... name.....	rate.per.second.....		total..... count.....	average..... per.trans....
	max.....	cur.....		
fetch for read	20	0	9.8	17821
fetch for write	47	0	1.2	2250
in LB: all ok	60	0	11.0	20031
LB: need lock	1	0	0.0	39
LB: old version	0	0	0.0	0
not found: read	0	0	0.0	1
: synth	0	0	0.0	0
DAPF: success	0	0	0.0	0
DAPF: failure	0	0	0.0	0
DAPF: utilized	0	0	0.0	0
DAPF: discarded	0	0	0.0	0

```
-----
```

```
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot Yank !
```

This problem has been corrected in Oracle Rdb7 Release 7.0.1.2. The RMU/SHOW STATISTICS utility has been enhanced with the "PIO Statistics-SPAM Access" screen. The purpose of this screen is to identify the reason why the SPAM was accessed, for either read or write.

Using the same database as the above example, consider the following screen:


```

-----
statistic..... rate.per.second..... total..... average.....
name..... max..... cur..... avg..... count..... per.trans....
fetch for read      20      0      9.3      17821      1.0
  uniform area scan  15      0      0.1       280       0.0
  record store fet   20      0      9.1      17541      1.0
  record modify fet   0      0      0.0        0       0.0
  record erase fet    0      0      0.0        0       0.0
fetch for write     47      0      1.1      2250       0.1
  record store upd    4      0      0.4       858       0.0
  record modify upd   0      0      0.0        0       0.0
  record erase upd    23     0      0.1       321       0.0
fetch for update    47      0      1.1      2250       0.1
  clump allocate      3      0      0.1       216       0.0
  fast incr. bkup     0      0      0.0        0       0.0
  threshold update    23     0      0.5       963       0.0
record stored       16      0      8.7      16677      1.0
record marked      1849     0     22.0     42049      2.5
record erased       622     0      4.3      8338       0.5
-----

```

Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot Yank !

As can be clearly seen, the majority of the SPAM page “fetch for read” accesses were caused by record storage. The SPAM page “fetch for write” accesses are evenly distributed between record stores and record erases.

Note that 16,677 records were stored, while 17,541 SPAM fetches occurred because of those stores. However, only 858 of those SPAM fetches actually resulted in updates to the SPAM thresholds.

Excessive SPAM fetches can be identified by comparing the “record store fet” field against the “record store upd” and “record stored” fields.

Table 7-6 “SPAM Access” Screen Fields

Field	Description
fetch for read	The total number of times the SPAM page was fetched for retrieval.
uniform area scan	The total number of times the SPAM page was fetched for retrieval during a record store operation. This is used primarily to check if SPAM thresholds need to be adjusted.
record store fet	The total number of times the SPAM page was fetched for retrieval during a record store operation. This is primarily used to check if SPAM thresholds need to be adjusted.
record modify fet	The total number of times the SPAM page was fetched for retrieval during a record modification operation. This is primarily used to check if SPAM thresholds need to be adjusted.
record erase fet	The total number of times the SPAM page was fetched for retrieval during a record erase operation. This is primarily used to check if SPAM thresholds need to be adjusted.

(continued on next page)

Table 7-6 (Cont.) “SPAM Access” Screen Fields

Field	Description
fetch for write	The total number of times the SPAM page was fetched for update.
record store upd	The total number of times the SPAM page was fetched for update during a record store operation. This is primarily used to modify the SPAM thresholds.
record modify upd	The total number of times the SPAM page was fetched for update during a record modification operation. This is primarily used to modify the SPAM thresholds.
record erase upd	The total number of times the SPAM page was fetched for update during a record erase operation. This is primarily used to modify the SPAM thresholds.
fetch for update	The total number of times the SPAM page was fetched for update.
clump allocate	The total number of times the SPAM page was updated for a clump allocation operation.
fast incr. bkup	The total number of times the SPAM page was updated for a fast incremental backup modification.
threshold update	The total number of times the SPAM page was updated to change a data page's threshold information.
record stored	The total number of records stored.
record marked	The total number of records modified.
record erased	The total number of records erased.

The “PIO Statistics–SPAM Access” screen is recorded to the binary output file, and is available during binary input file replay.

The following example shows the statistics collected following an operation that stored 8,192 records into an uniform-format storage area:

```
Node: ALPH (1/1/16) Oracle Rdb X7.0-00 Perf. Monitor 5-JAN-1998 12:20:06.57
Rate: 1.00 Second PIO Statistics--SPAM Access Elapsed: 00:10:42.88
Page: 1 of 1 DISK$:[WORK]MF_PERSONNEL.RDB;1 Mode: Online
```

```
-----
statistic..... rate.per.second..... total..... average.....
name..... max..... cur..... avg..... count..... per.trans....
fetch for read          19          0          13.6          8775          1.0
uniform area scan       1          0           0.1           110           0.0
record store fet       17          0          13.4          8665          1.0
record modify fet       0          0           0.0            0           0.0
record erase fet        0          0           0.0            0           0.0
fetch for write         3          0           0.9           642           0.0
record store upd        1          0           0.4           321           0.0
record modify upd       0          0           0.0            0           0.0
record erase upd        0          0           0.0            0           0.0
fetch for update        3          0           0.9           642           0.0
clump allocate          0          0           0.0            0           0.0
fast incr. bkup         0          0           0.0            0           0.0
threshold update        1          0           0.4           321           0.0
record stored           16          0          12.9          8338          1.0
record marked           17          0          13.4          8661          1.0
record erased           0          0           0.0            0           0.0
-----
```

```
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot
```

The following example shows the statistics collected following an operation that scanned 8,192 records into an uniform format storage area:

```
Node: ALPH (1/1/16) Oracle Rdb X7.0-00 Perf. Monitor 5-JAN-1998 12:42:44.65
Rate: 1.00 Second PIO Statistics--SPAM Access Elapsed: 00:01:25.56
Page: 1 of 1 DISK$:[WORK]MF_PERSONNEL.RDB;1 Mode: Online
```

```
-----
statistic..... rate.per.second..... total..... average.....
name..... max..... cur..... avg..... count..... per.trans....
fetch for read 158 0 3.8 330 165.0
uniform area scan 158 0 3.8 330 165.0
record store fet 0 0 0.0 0 0.0
record modify fet 0 0 0.0 0 0.0
record erase fet 0 0 0.0 0 0.0
fetch for write 0 0 0.0 0 0.0
record store upd 0 0 0.0 0 0.0
record modify upd 0 0 0.0 0 0.0
record erase upd 0 0 0.0 0 0.0
fetch for update 0 0 0.0 0 0.0
clump allocate 0 0 0.0 0 0.0
fast incr. bkup 0 0 0.0 0 0.0
threshold update 0 0 0.0 0 0.0
record stored 0 0 0.0 0 0.0
record marked 0 0 0.0 0 0.0
record erased 0 0 0.0 0 0.0
-----
```

Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot

The following example shows the statistics collected following an operation that modified 8,192 records into an uniform format storage area:

```
Node: ALPH (1/1/16) Oracle Rdb X7.0-00 Perf. Monitor 5-JAN-1998 12:24:44.15
Rate: 1.00 Second PIO Statistics--SPAM Access Elapsed: 00:03:34.91
Page: 1 of 1 DISK$:[WORK]MF_PERSONNEL.RDB;1 Mode: Online
```

```
-----
statistic..... rate.per.second..... total..... average.....
name..... max..... cur..... avg..... count..... per.trans....
fetch for read 543 0 31.4 6765 3382.5
uniform area scan 147 0 1.9 415 207.5
record store fet 529 0 29.5 6350 3175.0
record modify fet 0 0 0.0 0 0.0
record erase fet 0 0 0.0 0 0.0
fetch for write 63 0 3.3 711 355.5
record store upd 34 0 1.8 395 197.5
record modify upd 0 0 0.0 0 0.0
record erase upd 0 0 0.0 0 0.0
fetch for update 63 0 3.3 711 355.5
clump allocate 14 0 0.7 158 79.0
fast incr. bkup 0 0 0.0 0 0.0
threshold update 20 0 1.1 237 118.5
record stored 494 0 26.5 5703 2851.5
record marked 703 0 38.1 8192 4096.0
record erased 0 0 0.0 0 0.0
-----
```

Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot

The following example shows the number of SPAM pages retrieved in order to store a single, new record:

```

Node: ALPH (1/1/16) Oracle Rdb X7.0-00 Perf. Monitor 5-JAN-1998 12:25:55.48
Rate: 1.00 Second PIO Statistics--SPAM Access Elapsed: 00:00:24.01
Page: 1 of 1 DISK$:[WORK]MF_PERSONNEL.RDB;1 Mode: Online

```

```

-----
statistic..... rate.per.second..... total..... average.....
name..... max..... cur..... avg..... count..... per.trans....
fetch for read      720      0      31.4      756      756.0
  uniform area scan  180      0       7.9      190      190.0
  record store fet   539      0      23.5      566      566.0
  record modify fet    0      0       0.0       0       0.0
  record erase fet    0      0       0.0       0       0.0
fetch for write     1      0       0.0       2       2.0
  record store upd    0      0       0.0       1       1.0
  record modify upd   0      0       0.0       0       0.0
  record erase upd    0      0       0.0       0       0.0
fetch for update    1      0       0.0       2       2.0
  clump allocate      0      0       0.0       0       0.0
  fast incr. bkup     0      0       0.0       0       0.0
  threshold update    0      0       0.0       1       1.0
record stored       1      0       0.0       2       2.0
record marked       1      0       0.0       2       2.0
record erased       0      0       0.0       0       0.0

```

```

-----
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot

```

Now that the clump has been allocated, subsequent record storage into the same clump is significantly easier:

```

Node: ALPH (1/1/16) Oracle Rdb X7.0-00 Perf. Monitor 5-JAN-1998 12:27:05.47
Rate: 1.00 Second PIO Statistics--SPAM Access Elapsed: 00:00:36.53
Page: 1 of 1 DISK$:[WORK]MF_PERSONNEL.RDB;1 Mode: Online

```

```

-----
statistic..... rate.per.second..... total..... average.....
name..... max..... cur..... avg..... count..... per.trans....
fetch for read      621      0      17.8      653      326.5
  uniform area scan  156      0       4.4      164      82.0
  record store fet   465      0      13.3      489      244.5
  record modify fet    0      0       0.0       0       0.0
  record erase fet    0      0       0.0       0       0.0
fetch for write     0      0       0.0       0       0.0
  record store upd    0      0       0.0       0       0.0
  record modify upd   0      0       0.0       0       0.0
  record erase upd    0      0       0.0       0       0.0
fetch for update    0      0       0.0       0       0.0
  clump allocate      0      0       0.0       0       0.0
  fast incr. bkup     0      0       0.0       0       0.0
  threshold update    0      0       0.0       0       0.0
record stored       0      0       0.0       1       0.5
record marked       0      0       0.0       1       0.5
record erased       0      0       0.0       0       0.0

```

```

-----
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Unreset Write X_plot

```

7.8.11 RMU/SHOW STATISTICS Enhanced to Prevent Database Hangs

It was sometimes possible for the RMU/SHOW STATISTIC utility to cause the database to hang. This could occur when the utility was left idle at a user prompt or menu request.

The database would typically hang when opening or closing the database on a different node.

The following scenario shows the problem:

After rebooting a machine the operators tried to open all the databases and one of them wouldn't open. The RMU/OPEN MYDB command did not respond.

Typing the RMU/SHOW USERS MYDB command on the same node showed the following:

```
database MYDB.RDB;1
* database startup is in progress
* database is opened by an operator
* operator is waiting for reply to open request
```

Editing the monitor log for this node showed the equivalent of:

```
.
.
.
6-JAN-1998 11:08:00.01 - received open database request
from 202011C9:0
- process name _NTY150:, user JVAITKUN
- for database "MYDB" [_$1$DUA7] (34,75,0)
- database global section name is "RDM61T_3H300ND"
- database global section size is 172 pages (512 bytes per page)
- database startup waiting for MEMBIT lock
```

The database was open on all other nodes in the cluster but could not be connected to from the local node and the RMU/SHOW STATISTICS MYDB command would not work. Also, all attached processes appeared to be hung.

The problem was traced to a previously running RMU/SHOW STATISTICS screen that the operators had started. Operators were instructed to monitor the stall messages as well as to enable the alarm by pressing 'A'. In this instance they pressed 'S' for Set Rate and received the prompt "Enter time interval in seconds:" to which no one replied. As soon as a time was entered, the database opened.

The RMU/SHOW STATISTICS utility has been enhanced with the new command-line qualifier /PROMPT_TIMEOUT. This qualifier allows you to specify the user prompt timeout interval, in seconds. The default value is 60 seconds.

If you specify the /NOPROMPT_TIMEOUT qualifier or the value "0", the RMU/SHOW STATISTICS utility will not timeout any user prompts. Note that this is the current behavior and can potentially cause a database hang situation.

Note

Oracle recommends that you do not use the /NOPROMPT_TIMEOUT qualifier or the value "0" unless you are certain prompts will always be responded to in a timely manner.

If the /PROMPT_TIMEOUT qualifier is specified with a value less than ten seconds, the value "10" will be used.

The user prompt timeout interval can also be specified using the PROMPT_TIMEOUT configuration variable.

7.8.12 New SHOW STATISTICS Utility "AIJ Backup Activity" Screen

The RMU/SHOW STATISTICS utility has been enhanced with the new "AIJ Backup Activity" screen. Located in the "Process Information" sub-menu, the "AIJ Backup Activity" screen displays information about each AIJ backup operation being performed on the node.

The "AIJ Backup Activity" screen is also available during cluster-wide statistic collection. This means you can monitor the activities of all AIJ backup operations occurring on any node accessing the database.

The "AIJ Backup Activity" screen information is not recorded in the binary output file. Therefore, the screen is not available during binary file replay.

The following example shows a sample "AIJ Backup Activity" screen:

```
Node: ALPH (1/1/16) Oracle Rdb X7.0-00 Perf. Monitor 19-DEC-1997 15:50:24.49
Rate: 0.50 Seconds AIJ Backup Activity Elapsed: 00:03:57.99
Page: 1 of 1 DISK$: [WORK]MF_PERSONNEL.RDB;1 Mode: Online
```

```
-----
Process.ID Activity... VBN..... Operation..... Lock.ID.
34218467:1s block bkup 7:1017 Initializing AIJ journal
-----
```

The following example shows the same AIJ backup operation in a later stage of the backup operation:

```
Node: ALPH (1/1/16) Oracle Rdb X7.0-00 Perf. Monitor 19-DEC-1997 15:50:24.99
Rate: 0.50 Seconds AIJ Backup Activity Elapsed: 00:03:58.49
Page: 1 of 1 DISK$: [WORK]MF_PERSONNEL.RDB;1 Mode: Online
```

```
-----
Process.ID Activity... VBN..... Operation..... Lock.ID.
34218467:1s finish 7:1017 writing ROOT file (AIJFB VBN 1228)
-----
```

The "AIJ Backup Activity" screen contains five columns of information:

- **Process.ID:** This field contains the process identifier of the AIJ backup process. This process may be the AIJ backup server (ABS), in which case the process identifier will contain the suffix "s". This process may also be the manual RMU/BACKUP/AFTER_JOURNAL utility, in which case the process identifier will contain the "u" suffix.

Additional information can be obtained about this process by using the "Zoom" onscreen-menu option.

- **Activity:** This field contains a description of the backup activity being performed by the AIJ backup utility. The following backup activities are displayed:
 - **activation:** The AIJ backup utility is being invoked by the monitor if it is the ABS, or startup if the manual backup utility is being used.
 - **bind:** The AIJ backup utility is binding to the database.
 - **start:** The AIJ backup utility is starting the backup operation.
 - **create bkup:** The AIJ backup utility is creating a disk-based backup file.
 - **create temp:** The AIJ backup utility is creating a temporary AIJ journal. This activity typically occurs when the fast commit feature is used in conjunction with extensible AIJ journals.

- record bkup: The AIJ backup utility is backing up an extensible AIJ journal to disk, or any type of AIJ journal to tape, using a record-by-record transfer algorithm.
- block bkup: The AIJ backup utility is backing up a fixed-size (circular) AIJ journal to disk, using a 127-block transfer algorithm.
- finish: The AIJ backup utility is completing the backup of an AIJ journal.
- quiet-point: The AIJ backup utility is attempting to acquire the quiet-point lock.
- record shfl: The AIJ backup utility is performing the record shuffle operation used for extensible AIJ journals.
- unbind: The AIJ backup utility is unbinding from the database.
- VBN: This column identifies the current block number of the AIJ journal being backed up. The block number is normally prefixed with the AIJ sequence number, so it is easy to identify which AIJ journal is being backed up.
- Operation: This column identifies the activity-specific operation being performed by the AIJ backup utility. This column contains messages similar to those displayed by the “Stall Messages” screen.
- Lock.ID: This column identifies any lock the AIJ backup utility may be trying to acquire. This lock is typically the quiet-point lock.
More information about this lock can be obtained using the “LockID” onscreen-menu option.

7.9 Enhancements Provided in Oracle Rdb7 Release 7.0.1.1

7.9.1 Virtual Memory Statistics No Longer Collected

Oracle Rdb no longer collects virtual memory (VM) statistics and the information is no longer included in the RMU/SHOW STATISTICS utility. This information includes the RMU/SHOW STATISTICS items:

- GET_VM calls
- FREE_VM calls
- GET_VM kilobytes
- FREE_VM kilobytes
- \$EXPREG calls

7.9.2 New Logical Name RDMS\$CREATE_LAREA_NOLOGGING

This release of Oracle Rdb7 includes a new logical name which will disable journaling to the recovery unit journal (RUJ) and after-image journal (AIJ) during certain CREATE and ALTER operations.

Normally, when creating a new logical area as part of a CREATE TABLE, CREATE STORAGE MAP, CREATE INDEX, ALTER STORAGE MAP or ALTER INDEX statement, all updates to these logical areas are journalled to the recovery unit and after-image journals.

This can be a problem when creating or altering a large index, or reorganizing a storage map for a large table. In these cases, table rows, hash buckets, or B-tree nodes are written to the new logical areas and must be journalled to the RUJ and AIJ files. As the DDL operation proceeds, these records might also be re-journalled due to a subsequent update. The amount of I/O to these journals may

be extensive due to large amounts of I/O, and the long duration of the transaction may cause the RUJ and AIJ files to grow quite large.

In this release of Oracle Rdb7, this I/O to the recovery and after-image journals can be almost eliminated. The recovery and after-image journals will only contain a special logical operation with no associated data for the CREATE or ALTER operations. The savings during these DDL operations is less journaling I/O and low disk space requirements for these operations on large tables.

Note

Database administrators must be aware of the possible disadvantages of disabling journaling. The trade off is less I/O during the operation versus more complex recovery procedures. The changes in recovery are discussed below.

Using the RDMS\$CREATE_LAREA_NOLOGGING logical name

To disable journaling for these DDL commands the logical name RDMS\$CREATE_LAREA_NOLOGGING can be defined for the process which will perform the DDL operations. Once attached to the database all journaling to new logical areas will be disabled until the transaction which created them has been committed or rolled back.

The values accepted for this logical name are:

0—journaling is not disabled. This is the default if the logical name is not defined.

1—create logical area journaling is disabled. For example, on OpenVMS, the logical would be defined as shown in the following example.

```
$ DEFINE RDMS$CREATE_LAREA_NOLOGGING "1"
```

What happens if I need to recover my database using the after-image journal?

If after-image journaling is enabled when this logical name is in effect, then a warning message will be issued to inform the user that journaling was disabled during the transaction. The message recommends that a full database backup be performed as soon as possible. This is because the after-image journal no longer contains all the changes required to rebuild the new or altered database object. In effect, a hole or gap exists which means that the database can not be fully recoverable from the after-image journals.

```
$ DEFINE/USER RDMS$CREATE_LAREA_NOLOGGING 1
$ SQL$
ATTACH 'FILE DB$:TEST_NOJOURNAL';

CREATE INDEX T_I ON T (A);
%RDB-W-META_WARN, metadata successfully updated with the reported warning
-RDMS-W-DATACMIT, unjournalled changes made; database may not be recoverable
-RDMS-W-DOFULLBCK, full database backup should be done to ensure future
recovery

COMMIT;
```

When an after-image journal contains a record of an unjournalled create or alter and is used to rollforward the AIJ using the RMU/RECOVER command, then the logical area will be marked to indicate that it is incomplete. This can be seen using RMU/DUMP/LAREA=RDB\$AIP.


```

0003 0037 014F logical area 55, physical area 3
           06 0153 area name length 6 bytes
000000000000000000000000454C504D4953 0154 area name 'SIMPLE.....'
000000000000000000000000000000000000 0164 area name '.....'
           00000000 0173 snaps enabled TSN 0
           000D 0177 record length 13 bytes
00000000 0179 MBZ '....'
           0D 017D entry is in use
           0000 017E MBZ '..'
0000000 0180 thresholds are (0,0,0)
           00 0183 MBZ '.'
           area is corrupt and cannot be accessed

```

Oracle Rdb will report that the table or index is incomplete (has had unjournalled changes) if an attempt is made to use that table, or index after the recovery is complete. The only option at this time is to drop the table, storage map, or index and repeat the operation. In the following example the index T_I was created with logging disabled, this shows the results after the RMU/RECOVER command was used to recover the database.

```

SQL> SET FLAGS 'STRATEGY';
SQL> -- select using the incomplete index
SQL> -- FAILS
SQL> SELECT * FROM T WHERE A > 0;
Leaf#01 FFirst T Card=5
   BgrNdx1 T_I [1:0] Fan=17
%RDMS-F-DATATBLCMIT, unjournalled changes made to user-defined object
SQL>
SQL> -- now do a sequential scan
SQL> -- SUCCEEDS
SQL> SELECT B FROM T;
Get      Retrieval sequentially of relation T
          B
          NULL
          1
          NULL
          1
          2
5 rows selected
SQL>
SQL> -- now drop the index
SQL> -- SUCCEEDS
SQL> DROP INDEX T_I;
Firstn Get      Retrieval by index of relation RDB$INDICES
          Index name RDB$NDX_NDX_NAME_NDX [1:1]      Direct lookup
SQL>
SQL> -- select again (uses sequential scan)
SQL> -- SUCCEEDS
SQL> SELECT * FROM t WHERE A > 0;
Conjunct      Get      Retrieval sequentially of relation T
              A          B
              1          NULL
              1          1
              2          NULL
              2          1
              2          2
5 rows selected
SQL>
SQL> -- recreate the index
SQL> CREATE INDEX T_I ON T (A);
SQL>
SQL> -- select using the new index
SQL> -- SUCCEEDS
SQL> SELECT * FROM t WHERE A > 0;
Leaf#01 FFirst T Card=5

```

```

BgrNdx1 T_I [1:0] Fan=17
          A      B
          1      NULL
          1      1
          2      NULL
          2      1
          2      2
5 rows selected
SQL>
SQL> COMMIT;

```

Note

If CREATE INDEX or ALTER INDEX refers to a HASHED index then this operation also requires updates to the RDB\$SYSTEM_RECORD on each page of a MIXED format area. When these records are updated, the changes are always journalled to the recovery and after-image journals. Therefore, some journaling activity will result from these operations.

What if after-image journaling is disabled?

If after-image journaling is not currently in use, then the rollback operation can fully recover the database from the recovery unit journal. In this case, only the DATACMIT warning is issued. This is a warning that an error reported during the transaction must be rolled back to guarantee recovery. This is further discussed below.

What does RMU VERIFY report if one of the logical areas is marked incomplete?

RMU Verify will attempt to ready the incomplete logical area. The following example shows that the index T_I is incomplete (the warning message DATATBLCMIT) and the verify of the B-tree is abandoned.

```

$ RMU/VERIFY/ALL DB$:TEST_NOJOURNAL
.
.
.
%RMU-I-BGNNDXVER, beginning verification of index T_I
%RMU-W-DATATBLCMIT, unjournalled changes made to user-defined object
%RMU-E-BDLAREADY, error readying logical area with dbid 48
%RMU-W-NOT_LARDY, area for 48:560:0 not in proper ready mode
%RMU-E-BADDBKFET, Error fetching dbkey 48:560:0
%RMU-W-BTRVFYPRU, B-tree verification pruned at this dbkey
%RMU-I-BTRROODBK, root dbkey of B-tree is 48:560:0
%RMU-I-NDXERRORS,          2 index errors encountered
%RMU-I-ENDNDXVER, completed verification of index T_I
.
.
.

```

What happens if the CREATE or ALTER statement fails when journaling is disabled?

In this case, the creation of the logical area (which is always journalled) is rolled back. All the data written to that logical area is then erased from the database. To erase the data from a MIXED format area requires that each page of the storage area be processed. This will, most likely, be slower than similar recovery when journaling is enabled.

When recovery is performed using the `RMU/RECOVER` command, any rolled-back transaction is discarded (not applied to the backup database) and no reference to the incomplete logical area will be encountered.

What if an error occurs during the transaction?

If the transaction which performed the `CREATE` or `ALTER` statement has already committed, then subsequent transactions will have resumed journaling. This is the normal logging mode for Oracle Rdb and errors will be handled as expected.

However, if the original transaction which performed the `CREATE` or `ALTER` is still active and an error occurs while writing to an unjournalled logical area, then the logical area is immediately marked as corrupt. Such errors include failures of `INSERT` or `UPDATE` due to duplicate key values, constraint is violated, or database locking errors. The transaction should then be aborted using the `ROLLBACK` statement.

Although the `COMMIT` command can be used and the logical area deleted (using a `DROP` statement), this action may leave data anomalies which couldn't be rolled back at the time the error was detected. Oracle recommends that the transaction be rolled back.

Oracle recommends that the transaction be committed immediately after the `CREATE` or `ALTER` statement has successfully completed and avoid performing additional DML statements such as `INSERT`, `UPDATE` and `DELETE`. Committing promptly will avoid the problems described in this section and will also release locks on rows and other database system resources.

What happens if Hot Standby is active on the database?

The Hot Standby Option requires all operations to be journalled, therefore this logical name is ignored for any database enabled for Hot Standby.

Restriction for LIST STORAGE MAP

The disabling of logging is not supported when creating or altering a `LIST STORAGE MAP`. Please ensure that the `RDMSS$CREATE_LAREA_NOLOGGING` logical name is not defined when adding or making changes to a `LIST STORAGE MAP`. This also includes performing `IMPORT` operations which might implicitly create a `LIST STORAGE MAP`.

The reason for this restriction is that it is not possible for the rollback processing to distinguish between old and new `LIST` segments which might exist in the storage map. In Release 7.0.1.2 of Oracle Rdb7, the `RDMSS$CREATE_LAREA_NOLOGGING` logical name will be ignored during create or alter of a `LIST STORAGE MAP`.

7.9.3 Online Creation of Storage Areas Performed In Parallel

Similar to the `CREATE DATABASE MULTITHREAD AREA ADDITIONS` functionality, online storage area addition now initializes the pages of multiple storage areas in a multithreaded, or parallel, operation. Multithreaded storage area initialization permits multiple I/O operations to be issued to multiple devices, likely reducing the amount of time needed to create and initialize the storage areas.

In the following example, 10 new storage areas are created on 10 different disk devices. Assuming adequate process quotas, the 10 areas (the 5 live storage areas as well as the 5 snapshot storage areas) will be initialized with parallel I/O. This reduces the overall amount of time needed to initialize the storage areas.

```

SQL> ATTACH 'FILENAME MYDB';
SQL> ALTER DATABASE FILE MYDB
  ADD STORAGE AREA S1 FILENAME D1:[DB]S1 ALLOCATION 1000000
    SNAPSHOT FILENAME D6:[DB]S1 SNAPSHOT ALLOCATION 10000
  ADD STORAGE AREA S2 FILENAME D2:[DB]S2 ALLOCATION 1000000
    SNAPSHOT FILENAME D7:[DB]S2 SNAPSHOT ALLOCATION 10000
  ADD STORAGE AREA S3 FILENAME D3:[DB]S3 ALLOCATION 1000000
    SNAPSHOT FILENAME D8:[DB]S3 SNAPSHOT ALLOCATION 10000
  ADD STORAGE AREA S4 FILENAME D4:[DB]S4 ALLOCATION 1000000
    SNAPSHOT FILENAME D9:[DB]S4 SNAPSHOT ALLOCATION 10000
  ADD STORAGE AREA S5 FILENAME D5:[DB]S5 ALLOCATION 1000000
    SNAPSHOT FILENAME D0:[DB]S5 SNAPSHOT ALLOCATION 10000;

```

The multithreaded online storage area addition feature is enabled by default. To disable multithreaded online storage area additions, define the logical name `RDM$BIND_ONLINE_AREA_ADD_MULTITHREAD_COUNT` to "0". Off-line storage area addition does not utilize the multithreaded feature and continues to function as in previous versions of Oracle Rdb. Oracle recommends that you reserve storage area slots and then use online storage area addition.

By default, Oracle Rdb initializes up to 16 storage area files in parallel, and issues up to 2 write I/O requests per storage area at a time. The logical name `RDM$BIND_ONLINE_AREA_ADD_MULTITHREAD_COUNT` can be used to limit the number of storage areas that are initialized in parallel. Define this logical name to a value less than 128 to limit the number of files being initialized at once.

On OpenVMS, Oracle Rdb attempts to limit the number of parallel operations based on the process's remaining `FILLM`, `ASTLM` and `DIOLM` quotas. To ensure the highest level of performance, the recommended minimums for these process and system quotas for online area additions are listed in Table 7-7, Recommended Quota Minimums.

Table 7-7 Recommended Quota Minimums

Quota	Recommended Minimum
ASTLM	2 times the number of area files being added (including the snapshot storage area files), or 35, whichever is less.
DIOLM	2 times the number of area files being added (including the snapshot storage area files) or 35, whichever is less.
FILLM	At least enough available to open the additional number of storage area files being added (including the snapshot storage area files).
CHANNELCNT	At least enough available to open the additional number of storage area files being added (including the snapshot storage area files).
WSQUOTA	Large enough to avoid excessive page faulting. Each storage area being initialized in parallel requires at least an additional 400 working set pages on a VAX system or 25 working set pages on an Alpha system.

In general, utilizing more disk devices will result in increased performance when adding multiple storage areas. If you specify a large number of storage areas and many areas share the same device, a large multithread count could possibly cause excessive disk head movement, which may result in the storage area creation taking longer than if the areas were created one at a time. If this situation is the case, specify multiple `ALTER DATABASE...ADD STORAGE AREA` statements

or specify a smaller multithread count using the logical name RDM\$BIND_ONLINE_AREA_ADD_MULTITHREAD_COUNT.

7.9.4 Oracle7 Outer Join Syntax Support

Use of Oracle7 outer join syntax was not supported. Client applications originally written for Oracle7 might have used that syntax and failed. Now they should succeed.

The following example shows the Oracle7 outer join syntax.

```
SELECT * FROM A,B WHERE A.ACOL1(+) = B.BCOL1;
```

7.9.5 RMU/SHOW STATISTIC "Transaction Recovery Duration Estimate" Screen

One of the most difficult database attributes to determine is how long the database will be frozen if a process prematurely terminates, or how long a transaction rollback will take. Transaction recovery is affected by many factors, most of which are difficult to determine from runtime information available from the RMU/SHOW STATISTIC utility.

Therefore, the RMU/SHOW STATISTIC utility has been enhanced to provide an *estimate* of the time it will take to rollback a transaction, or to completely recover a failed process.

Disclaimer!

The information provided on the Transaction Recovery Duration Estimate screen is an estimate based on previous process recovery operations and other factors such as page contention and disk throughput.

However, it cannot be stressed enough that this information is an estimate only; the actual process recovery duration may be more or less than described on this screen.

Individual process failure recovery performance can vary widely depending on many factors which cannot be accounted for in the displayed estimate. These factors include lock deadlock stalls, network delays, disk contention and many other system factors such as lock remastering, etc.

The following example provides a sample transaction recovery scenario to consider:

```
Node: ALPH (1/1/2) Oracle Rdb X7.0-00 Perf. Monitor 17-AUG-1997 08:50:48.41
Rate: 1.00 Second Transaction Recovery Duration Estimate Elapsed: 00:29:04.34
Page: 1 of 1 DISK$: [WORK]MF_PERSONNEL.RDB;1 Mode: Online
```

```
-----
Process.ID RUJ.Sz Tx.Rollback DBR.Tx.Undo AIJ.Ckpt Pnd DBR.Tx.Redo DB.Freeze.Tm
2AA0ECC3:1 431 00:00:08.62 00:00:02.15 0:638 222 00:00:00.22 00:00:10.11
```

```
-----
Exit Help Menu >next_page <prev_page Refresh Set_rate Write Zoom !
```

The Transaction Recovery Duration Estimate screen provides the following information:

- **Process.ID** - This is the process identifier of a process that has the potential to rollback a transaction or require transaction recovery in the event of process failure.
- **RUJ.Sz** - This is the number of blocks of RUJ information that have been written by the process.
- **Tx.Rollback** - This is the *estimate* of the time it would require for the process to rollback the transaction. Note that this is different from the time it would take the DBR process to rollback the transaction.
- **DBR.Tx.Undo** - This is the *estimate* of the time it would require for the DBR process to “undo” the transaction. The DBR transaction undo duration is typically less than it takes the process to rollback the transaction, due to various optimizations and simplifications in the DBR recovery algorithm.
- **AIJ.Ckpt** - If the fast commit feature is enabled, this is the most recent checkpoint location in the AIJ journal for the process.
- **Pnd** - If AIJ journaling is enabled, this is the number of blocks of AIJ information that has been submitted (pending) but not yet written to the AIJ journal.
- **DBR.Tx.Redo** - If the fast commit feature is enabled, this is the *estimate* of the time it would take the DBR process to redo the failed process’ previously committed transactions to the database.
- **DB.Freeze.Tm** - This is the “estimate” of the total time the database would be frozen if the current process were to prematurely terminate.

In the above example, there are three estimates of essential information:

1. Process transaction rollback duration
2. DBR transaction undo and redo duration
3. Total database freeze duration

In the above example, if the process were to rollback the current transaction, it is estimated to take approximately 8 seconds. If the process were to fail prematurely, it is estimated to take the DBR process approximately 2 seconds to undo the transaction, but approximately .25 seconds to redo all previously committed transactions for that process. The total database freeze time is estimated to be approximately 10 seconds.

Validating the screen information can be performed by examining the end of the DBR logfile, which is enabled using the RDM\$BIND_DBR_LOG_FILE logical. For example:

```
18-AUG-1997 11:16:31.22 - TSN 0:291 was rolled back
18-AUG-1997 11:16:31.26 - Total recovery duration 10.10 seconds
```

Examining the past history of recovery operations can be performed using the RMU/DUMP/HEADER utility and reviewing the Database Recovery section. For example:

```

Database Recovery...
- 2 process failures have occurred (last 18-AUG-1997 11:16:31.26)
- DBR freeze averaging 5.470 seconds per recovery
  Transaction REDO averaging 0.890 seconds per recovery
  Transaction UNDO averaging 3.465 seconds per recovery
  AIJ recovery averaging 1.10 seconds per recovery
  Global buffer recovery averaging 0.0 seconds per recovery
  Global buffer tx recovery averaging 0.0 seconds per recovery
  Record cache recovery averaging 0.0 seconds per recovery
- DBR redo averaging 318 AIJ blocks per recovery
- DBR redo recovery rate averaging 2ms per AIJ block
- DBR undo averaging 635 RUJ blocks per recovery
- DBR undo recovery rate averaging 5ms per RUJ block
- DBR AIJ scan averaging 63 AIJ blocks per recovery
- DBR AIJ scan rate averaging 1ms per AIJ block
- Database is consistent but has been modified
- Full AIJ roll-forward is no longer permitted to this database
By-Area and By-Page AIJ roll-forward is permitted
- Full AIJ roll-forward to a newly restored database is permitted
- Next AIJ sequence number expected is 1
- Last commit transaction TSN is 0:320
- AIJ roll-forward is no-quiet-point enabled

```

The Transaction Recovery Duration Estimate screen is only available during online statistics collection. It is not available during binary file replay.

The configuration variable RECOVERY_SORT can be used to sort the Transaction Recovery Duration Estimate screen, by specifying one of the following keywords:

```

LONGEST_TRANSACTION - Sort by longest transaction rollback duration
LONGEST_UNDO - Sort by longest DBR undo duration estimate
LONGEST_REDO - Sort by longest DBR redo duration estimate
LONGEST_FREEZE - Sort by longest database freeze duration estimate

```

Of course, these sort criteria can also be selected online using the Config onscreen-menu option.

7.9.6 RMU/SHOW STATISTIC "File Overview" Sorting and Filtering Enhancements

The RMU/SHOW STATISTIC utility File IO Overview and File Lock Overview screens have been enhanced to provide additional sorting and filtering capabilities.

Two new sort options have been added to the screen configuration options, obtained using the Config onscreen-menu. The new Sort Alphabetically option sorts the storage area names without regards to storage area type (data or snapshot). The new Sort Alphabetically by Type option sorts the storage area names within storage area type (data or snapshot).

For example, the following File IO Overview screen shows the standard unsorted display:

Node: ALPH (1/1/2) Oracle Rdb X7.0-00 Perf. Monitor 25-AUG-1997 09:20:34.19
 Rate: 1.00 Second File IO Overview (Unsorted total I/O) Elapsed: 00:04:07.54
 Page: 1 of 1 DISK\$:[WORK]MF_PERSONNEL.RDB;1 Mode: Online

File/Storage.Area.Name.....	Sync.Reads	SyncWrites	AsyncReads	AsyncWrites	PgCkd
Database Root	17	0	0	1	0
AIJ (After-Image Journal)	0	0	0	0	0
RUJ (Recovery-Unit Journal)	0	0	0	0	0
ACE (AIJ Cache Electronic)	0	0	0	0	0
All data/snap files	3	0	0	0	0
data JOBS	0	0	0	0	0
data MF_PERS_DEFAULT	3	0	0	0	0
data SALARY_HISTORY	0	0	0	0	0
data DEPARTMENTS	0	0	0	0	0
data EMPIDS_LOW	0	0	0	0	0
data EMPIDS_MID	0	0	0	0	0
data EMPIDS_OVER	0	0	0	0	0
data EMP_INFO	0	0	0	0	0
data MF_PERS_SEGSTR	0	0	0	0	0
snap JOBS	0	0	0	0	0
snap MF_PERS_DEFAULT	0	0	0	0	0
snap SALARY_HISTORY	0	0	0	0	0
snap DEPARTMENTS	0	0	0	0	0
snap EMPIDS_LOW	0	0	0	0	0
snap EMPIDS_MID	0	0	0	0	0
snap EMPIDS_OVER	0	0	0	0	0
snap EMP_INFO	0	0	0	0	0
snap MF_PERS_SEGSTR	0	0	0	0	0

Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write

The following File IO Overview screen shows the display sorted alphabetically:

Node: ALPH (1/1/2) Oracle Rdb X7.0-00 Perf. Monitor 25-AUG-1997 09:20:40.92
 Rate: 1.00 Second File IO Overview (Alphabetical) Elapsed: 00:04:14.27
 Page: 1 of 1 DISK\$:[WORK]MF_PERSONNEL.RDB;1 Mode: Online

File/Storage.Area.Name.....	Sync.Reads	SyncWrites	AsyncReads	AsyncWrites	PgCkd
ACE (AIJ Cache Electronic)	0	0	0	0	0
AIJ (After-Image Journal)	0	0	0	0	0
All data/snap files	3	0	0	0	0
data DEPARTMENTS	0	0	0	0	0
snap DEPARTMENTS	0	0	0	0	0
Database Root	17	0	0	1	0
data EMPIDS_LOW	0	0	0	0	0
snap EMPIDS_LOW	0	0	0	0	0
data EMPIDS_MID	0	0	0	0	0
snap EMPIDS_MID	0	0	0	0	0
data EMPIDS_OVER	0	0	0	0	0
snap EMPIDS_OVER	0	0	0	0	0
data EMP_INFO	0	0	0	0	0
snap EMP_INFO	0	0	0	0	0
data JOBS	0	0	0	0	0
snap JOBS	0	0	0	0	0
data MF_PERS_DEFAULT	3	0	0	0	0
snap MF_PERS_DEFAULT	0	0	0	0	0
data MF_PERS_SEGSTR	0	0	0	0	0
snap MF_PERS_SEGSTR	0	0	0	0	0
RUJ (Recovery-Unit Journal)	0	0	0	0	0
data SALARY_HISTORY	0	0	0	0	0
snap SALARY_HISTORY	0	0	0	0	0

Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write

The following File IO Overview screen shows the display sorted alphabetically by type:

```
Node: ALPH (1/1/2) Oracle Rdb X7.0-00 Perf. Monitor 25-AUG-1997 09:20:44.42
Rate: 1.00 Second File IO Overview (Alphabetical) Elapsed: 00:04:17.77
Page: 1 of 1 DISK$:[WORK]MF_PERSONNEL.RDB;1 Mode: Online
```

File/Storage.Area.Name.....	Sync.Reads	SyncWrites	AsyncReads	AsyncWrites	PgCkd
ACE (AIJ Cache Electronic)	0	0	0	0	0
AIJ (After-Image Journal)	0	0	0	0	0
All data/snap files	3	0	0	0	0
Database Root	17	0	0	1	0
RUJ (Recovery-Unit Journal)	0	0	0	0	0
data DEPARTMENTS	0	0	0	0	0
data EMPIDS_LOW	0	0	0	0	0
data EMPIDS_MID	0	0	0	0	0
data EMPIDS_OVER	0	0	0	0	0
data EMP_INFO	0	0	0	0	0
data JOBS	0	0	0	0	0
data MF_PERS_DEFAULT	3	0	0	0	0
data MF_PERS_SEGSTR	0	0	0	0	0
data SALARY_HISTORY	0	0	0	0	0
snap DEPARTMENTS	0	0	0	0	0
snap EMPIDS_LOW	0	0	0	0	0
snap EMPIDS_MID	0	0	0	0	0
snap EMPIDS_OVER	0	0	0	0	0
snap EMP_INFO	0	0	0	0	0
snap JOBS	0	0	0	0	0
snap MF_PERS_DEFAULT	0	0	0	0	0
snap MF_PERS_SEGSTR	0	0	0	0	0
snap SALARY_HISTORY	0	0	0	0	0

```
Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write
```

Also, a new Filter onscreen-menu option has been added. The Filter onscreen-menu option prompts the user to enter a pattern string that includes wildcard characters. Using wildcard characters in the search pattern, for example, it is possible to find all EMP storage areas using the search pattern “*EMP*”.

Note

Search patterns specified without wildcard characters will find exact matches only. For example, the wildcard name “EMP” will find the single storage area whose name is “EMP”.

The pattern string may contain either one or both of the two wildcard characters, asterisk (*) and percent (%). The asterisk character is mapped to zero or more characters. The percent character is mapped to only one character.

You may enter a different filter for each screen.

Of course, filtering of the alphabetically sorted storage areas is permitted.

When a filter has been specified, the Filter onscreen-menu option will be highlighted. Selecting the Filter onscreen-menu option and pressing the RETURN key will delete any previously existing filter.

The following example shows the File IO Overview screen filtered using the pattern “*EMP*”:

```
Node: ALPH (1/1/2) Oracle Rdb X7.0-00 Perf. Monitor 25-AUG-1997 09:25:50.61
Rate: 1.00 Second File IO Overview (Unsorted total I/O) Elapsed: 00:00:05.57
Page: 1 of 1 DISK$:[WORK]MF_PERSONNEL.RDB;1 Mode: Online
```

File/Storage.Area.Name.....	Sync.Reads	SyncWrites	AsyncReads	AsyncWrits	PgCkd
data EMPIDS_LOW	0	0	0	0	0
data EMPIDS_MID	0	0	0	0	0
data EMPIDS_OVER	0	0	0	0	0
data EMP_INFO	0	0	0	0	0
snap EMPIDS_LOW	0	0	0	0	0
snap EMPIDS_MID	0	0	0	0	0
snap EMPIDS_OVER	0	0	0	0	0
snap EMP_INFO	0	0	0	0	0

```
Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write
```

Note

The “data” and “snap” prefixes are *not* part of the storage area name and are not considered when applying a specified filter. For example, the pattern “data*” will **NOT** find all data storage areas.

To control the selection of storage area types, three new sort options have been added to the screen configuration options, obtained using the Config onscreen-menu. The new Display All Storage Areas option displays all storage areas, as has previously been the case. The new Display Data Storage Areas Only option displays only live data storage areas. The new Display Snap Storage Areas Only option displays only snapshot storage areas.

The following example shows the File IO Overview screen displaying only live storage areas:

```
Node: ALPH (1/1/2) Oracle Rdb X7.0-00 Perf. Monitor 25-AUG-1997 13:46:22.48
Rate: 1.00 Second File IO Overview (Unsorted total I/O) Elapsed: 00:01:46.60
Page: 1 of 1 DISK$:[WORK]MF_PERSONNEL.RDB;1 Mode: Online
```

File/Storage.Area.Name.....	Sync.Reads	SyncWrites	AsyncReads	AsyncWrits	PgCkd
data MF_PERS_DEFAULT	3	0	0	0	0
data DEPARTMENTS	0	0	0	0	0
data EMPIDS_LOW	0	0	0	0	0
data EMPIDS_MID	0	0	0	0	0
data EMPIDS_OVER	0	0	0	0	0
data EMP_INFO	0	0	0	0	0
data JOBS	0	0	0	0	0
data MF_PERS_SEGSTR	0	0	0	0	0
data SALARY_HISTORY	0	0	0	0	0

```
Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write
```

The following example shows the File IO Overview screen displaying only snapshot storage areas:

```

Node: ALPH (1/1/2) Oracle Rdb X7.0-00 Perf. Monitor 25-AUG-1997 13:46:26.06
Rate: 1.00 Second File IO Overview (Unsorted total I/O) Elapsed: 00:01:50.18
Page: 1 of 1 DISK$:[WORK]MF_PERSONNEL.RDB;1 Mode: Online

```

```

-----
File/Storage.Area.Name..... Sync.Reads SyncWrites AsyncReads AsyncWrites PgCkd
snap MF_PERS_DEFAULT          0          0          0          0          0
snap DEPARTMENTS              0          0          0          0          0
snap EMPIDS_LOW                0          0          0          0          0
snap EMPIDS_MID                0          0          0          0          0
snap EMPIDS_OVER              0          0          0          0          0
snap EMP_INFO                  0          0          0          0          0
snap JOBS                      0          0          0          0          0
snap MF_PERS_SEGSTR            0          0          0          0          0
snap SALARY_HISTORY            0          0          0          0          0
-----

```

```

-----
Config Exit Filter Help Menu >next_page <prev_page Options Reset Set_rate Write
-----

```

7.9.7 RMU/SHOW STATISTIC Utility /OPTION=CONFIRM Qualifier

A new command qualifier has been added to the RMU/SHOW STATISTIC utility: /OPTION=CONFIRM. The CONFIRM keyword indicates that you wish to confirm before exiting from the utility.

This qualifier can also be specified in the configuration file using the CONFIRM_EXIT variable. A value of TRUE indicates you wish to confirm before exiting the utility, while a value of FALSE, the default value, indicates you do *not* want to confirm before exiting the utility.

7.9.8 RMU/SHOW STATISTIC Utility Fast Incremental Backup Display

The RMU/SHOW STATISTIC utility has been enhanced to display fast incremental backup runtime statistics in the Fast Incr Backup Statistics screen, located in the Journaling Information sub-menu.

The following is an example of the Fast Incr Backup Statistics screen:

```

Node: ALPH (1/1/2) Oracle Rdb X7.0-00 Perf. Monitor 11-SEP-1997 13:45:05.69
Rate: 0.50 Seconds Fast Incr Backup Statistics Elapsed: 00:35:38.17
Page: 1 of 1 DISK$:[WORK]MF_PERSONNEL.RDB;1 Mode: Online

```

```

-----
statistic..... rate.per.second..... total..... average.....
name..... max..... cur..... avg..... count..... per.trans....
FIB update attempt      32      0      10.3      22033      1.6
FIB map updated         0      0       0.0       15       0.0
SPAM page updated       0      0       0.0       15       0.0
SPAM updt deferred     32      0      10.2      22015      1.6
-----

```

```

-----
Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot Yank !
-----

```

The following explains the statistical information displayed:

- **FIB update attempt:** This statistic indicates the number of times the fast incremental backup (FIB) update operation was attempted. The *attempt* does not always result in the SPAM page being updated.
- **FIB map updated:** This statistic indicates the number of times the FIB map, a per-process data structure, was updated. This data structure indicates when each process no longer needs to update a particular SPAM page any longer.

- **SPAM page updated:** This statistic indicates the number of times a SPAM page was immediately modified to indicate that one or more pages in the SPAM interval have been modified since the last incremental backup. Each SPAM page update results in one synchronous read I/O and one synchronous write I/O operation.
- **SPAM updt deferred:** This statistic indicates the number of times a SPAM page did not need to be immediately modified, but might have to be modified at a later time. In most cases, this statistic closely follows the **FIB update attempt** statistic.

This screen is available during replay of a binary input file, and is also available cluster-wide.

7.9.9 RMU/SHOW STATISTIC Utility "Page Information" Zoom Screen

The RMU/SHOW STATISTIC utility has been integrated with the RMU/DUMP utility to provide runtime database page information displayed on a “zoom” screen. The page information is presented on a “zoom” screen in a format similar to that displayed by the RMU/DUMP/AREA=parea/START=pno/END=pno utility.

The page information zoom screen is currently available from the Stall Messages, Active User Stall Messages, DBR Activity and DBKEY Information screens.

The page information is selected using the PageInfo onscreen-menu option, by pressing the P key.

You will be prompted to select a process from the list of available processes with DBKEY information displayed on the screen. Only those processes displaying physical DBKEY information can be selected.

If the process you selected is accessing a range of pages, you will be prompted to select the desired page from a sub-menu provided.

If you are displaying page information from the DBKEY Information page, you will be prompted to select one of the types of pages being accessed by that process.

It is also possible to display an arbitrary page using the Tools menu, obtained using the exclamation point (!). Select the Display Page Information option and you will be prompted for the desired storage area and page number.

The following caveats apply to the page information display:

- For security reasons, the contents of individual lines on a data page *cannot* be displayed. The contents of area inventory pages cannot be displayed, either. Contact your DBA for other methods to display the contents of selected rows.
- Because the page information can be quite lengthy, you are able to migrate through the various pages using the “right-arrow” and “left-arrow” keys (1 page at a time) or the “up-arrow” and “down-arrow” keys (1 line at a time).
- Of course, the page information “zoom” screen contents can be written to disk using the Write onscreen-menu option (W key).
- The PageInfo onscreen-menu option is not available during replay of a binary input file.
- No locking of the selected page actually occurs. Therefore, it may be possible (but unlikely) to display inconsistent page information.

The PageInfo onscreen-menu option identifies and resolves logical DBKEYs and retrieves the corresponding physical DBKEYs.

Note

When using ALG, logical-DBKEYs such as "59:1:-3" are not resolveable, so SHOW STATISTICS retrieves the identified page, which in some cases is not always correct. In the above example, page "1" is a SPAM page, which obviously cannot be the target of the logical DBKEY.

The following is an example of a live data page information display:

```

-----+-----
0001 00000005 0000 page 5, physical area 1 (data)
      8EE86A99 0006 checksum = 8EE86A99
009BA463 0DA1FC74 000A time stamp = 14-SEP-1997 06:51:12.75
      0000 006A 0012 106 free bytes, 0 locked
      0002 0016 2 lines
      01AE 0240 0018 line 0: offset 0240, 430 bytes
      01AE 0092 001C line 1: offset 0092, 430 bytes
      0000018C 0020 line 0: TSN 396
      000001BF 0024 line 1: TSN 447
      00000024 03EE snap page pointer 36
      000001BF 03F2 snap pointer TSN 447
      003B 03F6 logical area 59
      0000003B 03F8 page sequence number 59
      0000 03FC page TSN base 0
      0000 03FE MBZ '..'
-----+-----

```

The following is an example of a snapshot data page information display:

```

-----+-----
4001 00000001 0000 page 1, physical area 1 (snap)
      A46ACD6A 0006 checksum = A46ACD6A
009BA304 993A8B26 000A time stamp = 12-SEP-1997 13:02:33.60
      0000 0054 0012 84 free bytes, 0 locked
      0003 0016 3 lines
      0000 0000 0018 line 0: empty
      01AE 0238 001C line 1: offset 0238, 430 bytes
      01AE 008A 0020 line 2: offset 008A, 430 bytes
      00000000 0024 line 0: TSN 0
      000085BD 0028 line 1: TSN 34237
      000085BF 002C line 2: TSN 34239
      0000 0030 line 0 -> live line: 0
      0000 0032 line 1 -> live line: 0
      0000 0034 line 2 -> live line: 0
      000001AB 03E6 live page pointer 427
      000085C4 03EA max TSN 34244
      FFFFFFFF 03EE snap page pointer -1
      00000000 03F2 snap pointer TSN 0
      0000 03F6 MBZ '..'
      00000000 03F8 page sequence number 0
      0000 03FC page TSN base 0
      0000 03FE MBZ '..'
-----+-----

```

The following is an example of an area inventory page (AIP) page information display:

```

0001 000001D0 0000 page 464, physical area 1 (AIP)
          D992664E 0006 checksum = D992664E
009646FF 8BFE44E0 000A time stamp = 1-DEC-1992 12:49:48.
          0000 0022 0012 34 free bytes, 0 locked
          000001D1 0016 next area inventory page 465
          4001 03F6 logical area 16385
          00000000 03F8 page sequence number 0
          0000 03FC page TSN base 0

```

The following is an example of a SPAM page information display; note that a SPAM page display is quite lengthy:

```

0001 00000001 0000 page 1, physical area 1 (SPAM)
          4681E156 0006 checksum = 4681E156
      80000000 00000060 000A Fast incremental backup TSN = 0:96
          0000 0001 0012 1 free byte, 0 locked
FFFFFFFFFFFFFFFFCFFFFFFFFFFFFFFFFF 0016 pages 2-31: threshold 3
          page 32: threshold 0
          pages 33-65: threshold 3
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 0026 pages 66-129: threshold 3
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 0036 pages 130-193: threshold 3
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 0046 pages 194-257: threshold 3
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF 0056 pages 258-321: threshold 3
0FFFFFFFC33C3FFFFFFFFFFFFFFFFFFFF 0066 pages 322-362: threshold 3
          pages 363-364: threshold 0
          pages 365-366: threshold 3
          page 367: threshold 0
          page 368: threshold 3
          pages 369-370: threshold 0
          pages 371-383: threshold 3
          pages 384-385: threshold 0
33FFFFFFFF03FFFFFFC03FFFFF3FF0FFFF 0076 pages 386-395: threshold 3
          pages 396-397: threshold 0
          pages 398-402: threshold 3
          page 403: threshold 0
          pages 404-414: threshold 3
          pages 415-418: threshold 0
          pages 419-430: threshold 3
          pages 431-433: threshold 0
          pages 434-446: threshold 3
          page 447: threshold 0
          page 448: threshold 3
          page 449: threshold 0
          page 513: threshold 0
.
.
.
0052 03E7 pages 1055-1057, logical area 82
0052 03E9 pages 1058-1060, logical area 82
0052 03EB pages 1061-1063, logical area 82
0052 03ED pages 1064-1066, logical area 82
0052 03EF pages 1067-1069, logical area 82
0052 03F1 pages 1070-1072, logical area 82
0052 03F3 pages 1073-1075, logical area 82
0052 03F5 pages 1076-1078, logical area 82
0052 03F7 pages 1079-1081, logical area 82
0052 03F9 pages 1082-1084, logical area 82
0052 03FB pages 1085-1087, logical area 82
0052 03FD pages 1088-1090, logical area 82
          00 03FF MBZ free '.'

```

7.9.10 RMU/SHOW STATISTIC "Logical Area" Menu Filter Option

Using the RMU/SHOW STATISTIC utility Logical Area menu was difficult when production databases contained hundreds or thousands of logical areas. One database required accessing the MORE option 230 times to get to the desired logical area.

The contents of the logical area menu can now be controlled by the use of wildcard selection criteria.

A new option has been added to the Tools menu, obtained using the exclamation mark (!) from any screen. The new Logical Area Menu Filter option lets you specify a search pattern containing wildcards.

Note

The specified pattern *MUST* match at least one logical area, or the pattern will be rejected.

The filtered logical area menu is only available when displaying all logical areas. It is *not* available if you selected the Display Application Logical Areas option from the Tools menu.

The specified pattern string may contain either one or both of the two wildcard characters, asterisk (*) and percent (%). The asterisk character is mapped to zero or more characters. The percent character is mapped to only one character. For example, the pattern “*EMP*” will find *any* logical area containing the text “EMP”, while the pattern “EMP*” will find only those logical areas whose name starts with “EMP”.

7.9.11 RMU/SHOW STATISTIC "Stall Messages" Screen Allows Wildcards

The RMU/SHOW STATISTIC utility Stall Messages screen Filter onscreen-menu option now allows the use of wildcards in the filtering criteria.

The pattern string may contain either one or both of the two wildcard characters, asterisk (*) and percent (%). The asterisk character is mapped to zero or more characters. The percent character is mapped to only one character.

7.9.12 CPU Time Displayed Correctly

Previously, the Oracle Rdb RMU/SHOW STATISTICS interface was unable to correctly display process CPU times in excess of 1 day; the number of days value was not displayed.

Oracle Rdb RMU/SHOW STATISTICS is now able to display CPU times greater than one day. Because the width of the CPU time display is limited, the following CPU time display formats are used:

- For CPU time values less than 1 day: “HH:MM:SS.CC”
- For CPU time values less than 100 days but more than 1 day: “DD HH:MM”
- For CPU time values more than 100 days: “DDD HH:MM”

LogMiner for Rdb

Oracle Rdb after-image journal (.aij) files contain a wealth of useful information about the history of transactions in a database. After-image journal files contain all of the data needed to perform database recovery. These files record every change made to data and metadata in the database. The LogMiner for Rdb feature provides an interface to the data record contents of Oracle Rdb after-image journal files. Data records that are added, updated, or deleted by committed transactions may be extracted (unloaded) from the .aij files in a format suitable for subsequent loading into another database or for use by user-written application programs.

Oracle Rdb after-image journaling protects the integrity of your data by recording all changes made by committed transactions to a database in a sequential log or journal file. Oracle Corporation recommends that you enable after-image journaling to record your database transaction activity between full backup operations as part of your database restore and recovery strategy. The after-image journal file is also used to enable several database performance enhancements (such as the fast commit, row cache, and hot standby features).

See the *Oracle Rdb7 Guide to Database Maintenance* for more information about setting up after-image journaling.

To use LogMiner for Rdb, follow these steps:

1. Enable the database for LogMiner operation using the RMU Set Logminer command. See Section 8.1 for additional information.
2. Back up the after-image journal file using the Quiet_Point qualifier to the RMU Backup command.
3. Extract changed records using the RMU Unload After_Journal command. See Section 8.2 for additional information.

8.1 RMU Set Logminer Command

Allows you to change the LogMiner state of a database.

Format

RMU/Set Logminer root-file-spec

Command Qualifiers

/Disable
/Enable
/[No]Log

Defaults

See description
See description
Current DCL verify value

Description

Use this command to enable or disable LogMiner operations on an Oracle Rdb database. When LogMiner is enabled, the Oracle Rdb database software writes additional information to the after-image journal file when records are added, modified, and deleted from the database. This information is used during the unload operation.

Command Parameters

root-file-spec

The root file specification of the database. The default file extension is .rdb.

Command Qualifiers

Disable

Specifies that LogMiner operations are to be disabled for the database. When LogMiner is disabled, the Oracle Rdb software does not journal information required for LogMiner operations. When LogMiner is disabled for a database, the RMU Unload After_Journal command is not functional on that database.

Enable

Specifies that LogMiner operations are to be enabled for the database. When LogMiner is enabled, the Oracle Rdb database software logs additional information to the after-image journal file. This information allows LogMiner to extract records. The database must already have after-image journaling enabled.

Log

Nolog

Specifies that the setting of the LogMiner state for the database be reported to SYS\$OUTPUT. The default is the setting of the DCL VERIFY flag, which is controlled by the DCL SET VERIFY command.

Usage Notes

- To use the RMU Set Logminer command, you must have the RMU\$BACKUP, RMU\$RESTORE, or RMU\$ALTER privilege in the root file access control list (ACL) for the database or the OpenVMS SYSPRV or BYPASS privilege.
- The RMU Set Logminer command requires offline access to the database. The database must be closed and no other users may be accessing the database.
- Execute a full database backup operation after issuing an RMU Set Logminer command that displays the RMU-W-DOFULLBCK warning message. Immediately after enabling LogMiner, you should perform a database after-image journal backup using the RMU Backup After_Journal command.

Examples

Example 1

The following example enables a database for LogMiner for Rdb operation.

```
$ RMU /SET LOGMINER /ENABLE OLTPDB.RDB
```

8.2 RMU Unload After_Journal Command

Allows you to extract added, modified, and deleted record contents from committed transactions from specified tables in one or more after-image journal files.

Format

RMU/Unload/After_Journal root-file-spec aij-file-name

Command Qualifiers

/Before=date-time
/Extend_Size=integer
/IO_Buffers=integer
/[No]Log
/Options=File=file-spec
/Output=file-spec
/Select=selection-type
/Since=date-time
/Sort_Workfiles=integer
/Statistics_Interval=integer
/Table=(Name=table-name, [table-options ...])
/[No]Trace

Defaults

None
/Extend_Size=1000
/IO_Buffers=2
Current DCL verify value
See description
/Output=SYS\$OUTPUT
/Select=Commit_Transaction
None
/Sort_Workfiles=2
See description
See description
/Notrace

Description

The RMU Unload After_Journal command translates the binary data record contents of an after-image journal (.aij) file into an output file. Data records for the specified tables for committed transactions are extracted to an output stream (file, device, or application callback) in the order that the transactions were committed.

To use the RMU Unload After_Journal command, you must have first enabled the database for LogMiner extraction. Use the RMU Set Logminer command to enable the LogMiner for Rdb feature for the database. See the Section 8.1 for more information.

Data records extracted from the .aij file are those records that transactions added, modified, or deleted in base database tables. Index nodes, database metadata, segmented strings (BLOB), views, COMPUTED BY columns, system records, and temporary tables cannot be unloaded from after-image journal files.

Only the final content of a record for each transaction is extracted. Multiple changes to a single record within a transaction are condensed so that only the last revision of the record appears in the output stream. It is not possible to determine which columns were changed in a data record directly from the after-image journal file. The record itself would have to be compared to the content of a previous record in order to determine which columns were changed.

The database used to create the after-image journal files being extracted must be available during the RMU Unload After_Journal command execution. The database is used to obtain metadata information (such as table names, column counts, record version, and record compression) needed to extract data records from the .aij file. The database may be accessed either locally (on the same computer system) or remotely (over a network connection). The database is used

only as a metadata reference. The database is read solely to load the metadata and is then detached.

The after-image journal file or files are processed sequentially, and all specified tables are extracted in one pass through the after-image journal file.

As each transaction commit record is processed, all modified and deleted records for the specified tables are sorted to remove duplicates and then the modified and deleted records are written to the output streams. Transactions that were rolled back are discarded. Data records for tables not being extracted are discarded. The actual order of output records within a transaction is not predictable.

In the extracted output, records that were modified or added are returned as being modified. It is not possible to distinguish between inserted and updated records in the output stream. Deleted (erased) records are returned as being deleted. A transaction that modifies and deletes a record generates only a deleted record. A transaction that adds a new record to the database and then deletes it within the same transaction generates only a deleted record.

LogMiner signals that a row has been deleted by placing a D in the RDB\$LM_ACTION field and then recording the contents of the row at the instant before the delete operation in the user fields of the output record. If a row was modified several times within a transaction before being deleted, the output record will contain only the delete indicator and the results of the last modify operation. If a row is inserted and deleted in the same transaction, only the delete record appears in the output.

Records from multiple tables may be output to the same or to different destination streams. Possible output destination streams include the following:

- File
- OpenVMS Mailbox
- OpenVMS Pipe
- Direct callback to an application through a run-time activated sharable image

Command Parameters

root-file-spec

The root file specification of the database for the after-image journal file from which tables will be unloaded. The default file extension is .rdb.

The database must be the same database that was used to create the after-image journal files. The database is required so that the table metadata (information about data) is available to the RMU Unload After_Journal command. In particular, the names and relation identification of valid tables within the database is required along with the number of columns in the table and the compression information for the table in various storage areas.

The process attaches to the database briefly at the beginning of the extraction operation in order to read the metadata. Once the metadata has been read, the process disconnects from the database for the remainder of the operation.

aij-file-name

One or more input after-image journal backup files to be used as the source of the extraction operation. Multiple journal files can be extracted by specifying a comma-separated list of file specifications. OpenVMS wildcard specifications (using the * and % characters) are supported to extract a group of files. A

file specification beginning with the at (@) character refers to an options file containing a list of after-image journal files (rather than the file specification of an after-image journal itself). If you use the at (@) character syntax, you must enclose the at (@) character and the file name in double quotation marks (for example, specify `aij-file-name` as `"@files.opt"`). The default file extension is `.aij`.

Command Qualifiers

Before=date-time

Specifies the ending time and date for transactions to be extracted. Based on the Select qualifier, transactions that committed or started prior to the specified Before date are selected. Information changed due to transactions that committed or started after the Before date is not included in the output.

Extend_Size=integer

Specifies the file allocation and extension quantity for the output data files. The default extension size is 1000 blocks. Using a larger value can help reduce output file fragmentation and can improve performance when large amounts of data are extracted.

IO_Buffers=integer

Specifies the number of I/O buffers used for the output data files. The default number of buffers is 2. The default value is generally adequate. With sufficiently fast I/O subsystem hardware, additional buffers may improve performance. However, using a larger number of buffers will also consume additional virtual memory and process working set.

Log

Nolog

Specifies that the extraction of the `.aij` file be reported to `SYSS$OUTPUT` or the destination specified with the Output qualifier. When activity is logged, the output from the Log qualifier provides the number of transactions committed and rolled back. The default is the setting of the DCL VERIFY flag, which is controlled by the DCL SET VERIFY command.

Options=File=file-spec

An options file contains a list of tables and output destinations. The options file may be used instead of, or along with, the Table qualifier to specify the tables to be extracted. Each line of the options file must specify a table name prefixed with `"Table="`. After the table name, the output destination is specified as either `"Output="` or `"Callback_Module="` and `"Callback_Routine="`.

```
TABLE=tblname,CALLBACK_MODULE=image,CALLBACK_ROUTINE=routine  
TABLE=tblname,OUTPUT=outfile
```

The `Record_Definition=file-spec` option from the Table qualifier can be used to create a record definition file for the output data. The default file type is `.rrd` and the default file name is the name of the table.

The `Table_Definition=file-spec` option from the Table qualifier can be used to create a file with an SQL statement to create a table to hold transaction data. The default file type is `.sql` and the default file name is the name of the table.

Each option in the Options=File qualifier must be fully specified (no abbreviations are allowed) and followed with an equal sign (=) and a value string. The value string must be followed by a comma or the end of a line. Continuation lines may be specified by using a trailing dash. Comments are indicated by using the exclamation point (!) character.

Output=file-spec

Redirects the log and trace output (selected with the Log and Trace qualifiers) to the named file. If this qualifier is not specified, the output generated by the Log and Trace qualifiers, which can be voluminous, is displayed to SYSS\$OUTPUT.

Select=selection-type

Specifies if the date and time of the Before and Since qualifiers refers to transaction start time or transaction commit time.

The following options can be specified as the selection-type of the Select qualifier:

- **Commit_Transaction**
Specifies that the Before and Since qualifiers select transactions based on the time of the transaction commit.
- **Start_Transaction**
Specifies that the Before and Since qualifiers select transactions based on the time of the transaction start.

The default for date selection is **Commit_Transaction**.

Since=date-time

Specifies the starting time for transactions to be extracted. Based on the Select qualifier, transactions that committed on or after the specified Since date are selected. Information from transactions that committed or started prior to the specified Since date is not included in the output.

Sort_Workfiles=integer

Specifies the number of sort work files. The default number of sort work files is 2. When large transactions are being extracted, using additional sort work files may improve performance by distributing I/O loads over multiple disk devices. Use the SORTWORKn (where n is a number from 0 to 9) logical names to specify the location of the sort work files.

Statistics_Interval=integer

Specifies that statistics are to be displayed at regular intervals so that you can evaluate the progress of the unload operation.

The displayed statistics include:

- Elapsed time
- CPU time
- Buffered I/O
- Direct I/O
- Page faults
- Number of records unloaded for a table

If the **Statistics_Interval** qualifier is specified, the default interval is 60 seconds (1 minute). The minimum value is 1 second. If the unload operation completes successfully before the first time interval has passed, you will receive an informational message on the number of files unloaded. If the unload operation is unsuccessful before the first time interval has passed, you will receive error messages and statistics on the number of records unloaded.

At any time during the unload operation, you can press Ctrl/T to display the current statistics.

Table=(Name=table-name, table-options)

Specifies the name of a table to be unloaded and an output destination. The table-name must be a table within the database. Views, indexes, and system tables may not be unloaded from the after-image journal file.

The following table-options can be specified with the Table qualifier:

- **Output=file-spec**
If an Output file specification is present, unloaded records are written to the specified location.
- **Callback_Module=image-name, Callback_Routine=routine-name**
LogMiner for Rdb uses the OpenVMS library routine LIB\$FIND_IMAGE_SYMBOL to activate the specified sharable image and locate the specified entry point routine name. This routine will be called with each extracted record. A final call is made with the "Action" field set to "E" to indicate the end of the output stream. These options must be specified together.
- **Record_Definition=file-spec**
The Record_Definition=file-spec option can be used to create a record definition .rrd file for the output data. The default file type is .rrd and the default file name is the name of the table.
- **Table_Definition=file-spec**
The Table_Definition=file-spec option can be used to create a file with an SQL statement to create a table to hold transaction data. The default file type is .sql and the default file name is the name of the table.

Note that, unlike other qualifiers where only the final occurrence of the qualifier is used by an application, the Table qualifier may be specified multiple times for the RMU Unload After_Journal command. Each occurrence of the Table qualifier must specify a different table.

Trace**NoTrace**

Specifies that the unloading of the .ajj file be traced. The default is Notrace. When the unload operation is traced, the output from the Trace qualifier identifies transactions in the .ajj file by transaction sequence numbers (TSNs) and describes what Oracle RMU did with each transaction during the unload process. You can specify the Log qualifier with the Trace qualifier.

Usage Notes

- To use the RMU Unload After_Journal command for a database, you must have the RMU\$DUMP privilege in the root file access control list (ACL) for the database or the OpenVMS SYSPRV or BYPASS privilege.
- You can only extract changed records from a backup copy of the after-image journal files. You create this file using the RMU Backup After_Journal command. You also cannot extract from an .ajj file that has been optimized with the RMU Optimize After_Journal command. And, you cannot extract an active, primary .ajj file.

- As part of the extraction process, Oracle RMU sorts extracted journal records to remove duplicate record updates. Because .aij file extraction uses the OpenVMS Sort/Merge Utility (SORT/MERGE) to sort journal 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 Sort_Workfiles qualifier of the RMU Unload After_Journal command. The allowed values are 1 through 10 inclusive, with a default value of 2. The location of these work files can be specified with device specifications, using the SORTWORKn logical name (where n is a number from 0 to 9). See the OpenVMS documentation set for more information on using SORT/MERGE. See the *Oracle Rdb7 Guide to Database Performance and Tuning* for more information on using these Oracle Rdb logical names.
- You can redirect the .aij rollforward temporary work files to a different disk and directory location than the current default directory by assigning a different directory to the RDM\$BIND_AIJ_WORK_FILE logical name in the LNM\$FILE_DEV name table. This can help to alleviate I/O bottlenecks that might occur on the default disk.
- The RMU Unload After_Journal command can read either a backed up .aij file on disk or a backed up .aij file on tape that is in the Old_File format.
- One or more tables can be selected to be extracted from an after-image journal file. All tables specified by the Table qualifier and all those specified in the Options file are combined to produce a single list of output streams. A particular table may be specified only once. Multiple tables may be written to the same output destination by specifying the exact same output stream specification (that is, by using an identical file specification).
- At the completion of the unload operation, RMU creates a number of DCL symbols that contain information about the extraction statistics. For each table extracted, RMU creates the following symbols:
 - RMU\$UNLOAD_DELETE_COUNT_tablename
 - RMU\$UNLOAD_MODIFY_COUNT_tablename
 - RMU\$UNLOAD_OUTPUT_tablename

The tablename component of the symbol is the name of the table. When multiple tables are extracted in one operation, multiple sets of symbols are created. The value for the symbols RMU\$UNLOAD_MODIFY_COUNT_tablename and RMU\$UNLOAD_DELETE_COUNT_tablename is a character string containing the number of records returned for modified and deleted rows. The RMU\$UNLOAD_OUTPUT_tablename symbol is a character string indicating the full file specification for the output destination, or the sharable image name and routine name when the output destination is an application callback routine.

- When using the Callback_Module and Callback_Routine option, you must supply a sharable image with a universal symbol or entry point for LogMiner to be able to call your routine. See the OpenVMS manual discussing the Linker utility for more information about creating sharable images.
- Your Callback_Routine will be called once for each output record. The Callback_Routine will be passed two parameters:
 - The length of the output record, by longword value
 - A pointer to the record buffer

The record buffer is a data structure of the same fields and lengths written to an output destination.

- Because the Oracle RMU image is a known image, your sharable image must also be a known image. Use the OpenVMS Install Utility to make your sharable image known. You may wish to establish an exit handler to perform any required cleanup processing at the end of the extraction.

Examples

Example 1

The following example unloads the EMPLOYEES table from the .aij backup file MFP.AIJBCK.

```
RMU /UNLOAD /AFTER_JOURNAL MFP.RDB MFP.AIJBCK -  
  /TABLE = (NAME = EMPLOYEES, OUTPUT = EMPLOYEES.DAT)
```

Example 2

The following example simultaneously unloads the SALES, STOCK, SHIPPING, and ORDERS tables from the .aij backup files MFS.AIJBCK_1-JUL-1999 through MFS.AIJBCK_3-JUL-1999. Note that the input .aij backup files are processed sequentially in the order specified.

```
$ RMU /UNLOAD /AFTER_JOURNAL MFS.RDB -  
  MFS.AIJBCK_1-JUL-1999, -  
  MFS.AIJBCK_2-JUL-1999, -  
  MFS.AIJBCK_3-JUL-1999 -  
  /TABLE = (NAME = SALES, OUTPUT = SALES.DAT) -  
  /TABLE = (NAME = STOCK, OUTPUT = STOCK.DAT) -  
  /TABLE = (NAME = SHIPPING, OUTPUT = SHIPPING.DAT) -  
  /TABLE = (NAME = ORDER, OUTPUT = ORDER.DAT)
```

Example 3

To unload data based on a time range, use the Before and Since qualifiers. The following example extracts changes made to the PLANETS table by transactions that committed between 1-SEP-1999 at 14:30 and 3-SEP-1999 at 16:00.

```
$ RMU /UNLOAD /AFTER_JOURNAL MFS.RDB MFS.AIJBCK -  
  /TABLE = (NAME = PLANETS, OUTPUT = PLANETS.DAT) -  
  /BEFORE = "3-SEP-1999 16:00:00.00" -  
  /SINCE = "1-SEP-1999 14:30:00.00"
```

Example 4

The following example simultaneously unloads the SALES and STOCK tables from all .aij backup files that match the wildcard specification MFS.AIJBCK_1999-07-*. The input .aij backup files are processed sequentially in the order returned from the file system.

```
$ RMU /UNLOAD /AFTER_JOURNAL MFS.RDB -  
  MFS.AIJBCK_1999-07-* -  
  /TABLE = (NAME = SALES, OUTPUT = SALES.DAT) -  
  /TABLE = (NAME = STOCK, OUTPUT = STOCK.DAT)
```

Example 5

The following example unloads the TICKER table from the .aij backup files listed in the file called AIJ_BACKUP_FILES.DAT (note the double quotation marks surrounding the at (@) character and the file specification). The input .aij backup files are processed sequentially. The output records are written to the mailbox

device called MBA127:. A separate program is already running on the system, and it reads and processes the data written to the mailbox.

```
$ RMU /UNLOAD /AFTER_JOURNAL MFS.RDB -  
  "@AIJ_BACKUP_FILES.DAT" -  
  /TABLE = (NAME = TICKER, OUTPUT = MBA127:)
```

Example 6

To move transaction data from one database into a change table in another database, you can use the RMU Unload After_Journal command followed by RMU Load commands. A record definition (.rrd) file would need to be created for each table being loaded into the target database. The record definition files can be created by specifying the Record_Definition option on the Table qualifier.

```
$ RMU /UNLOAD /AFTER_JOURNAL OLTP.RDB MYAIJ.AIJBCK -  
  /TABLE = ( NAME = MYTBL, -  
            OUTPUT = MYTBL.DAT, -  
            RECORD_DEFINITION=MYLOGTBL) -  
  /TABLE = ( NAME = SALE, -  
            OUTPUT=SALE.DAT, -  
            RECORD_DEFINITION=SALELOGTBL)  
  
$ RMU /LOAD WAREHOUSE.RDB MYLOGTBL MYTBL.DAT -  
  /RECORD_DEFINITION = FILE = MYLOGTBL.RRD  
  
$ RMU /LOAD WAREHOUSE.RDB SALELOGTBL SALE.DAT -  
  /RECORD_DEFINITION = FILE = SALELOGTBL.RRD
```

Example 7

Instead of the Table qualifier, an Options file can be used to specify the table or tables to be extracted, as shown in the following example.

```
$ TYPE TABLES.OPTIONS  
TABLE=MYTBL, OUTPUT=MYTBL.DAT  
TABLE=SALES, OUTPUT=SALES.DAT  
$ RMU /UNLOAD /AFTER_JOURNAL OLTP.RDB MYAIJ.AIJBCK -  
  /OPTIONS = FILE = TABLES.OPTIONS
```

8.3 Restrictions and Limitations with LogMiner for Rdb

The following restrictions exist for the LogMiner for Rdb feature:

- Temporary tables cannot be extracted. Modifications to temporary tables are not written to the after-image journal file and, therefore, are not available to LogMiner for Rdb.
- Optimized after-image journal files cannot be used as input to the LogMiner for Rdb. Information needed by the RMU Unload After_Journal command is removed by the optimization process.
- Records removed from tables using the SQL TRUNCATE TABLE statement are not extracted. The SQL TRUNCATE TABLE statement does not journal each individual data record being removed from the database.
- Records removed by dropping tables using the SQL DROP TABLE statement are not extracted. The SQL DROP TABLE statement does not journal each individual data record being removed from the database.
- Tables that use the vertical record partitioning (VRP) feature cannot be extracted using LogMiner for Rdb. LogMiner software currently does not detect these tables. A future release of Oracle Rdb will detect and reject access to vertically partitioned tables.
- Segmented string data (BLOB) cannot be extracted using LogMiner for Rdb. Because the segmented string data is related to the base table row by means of a database key, there is no convenient way to determine what data to extract. Additionally, the data type of an extracted column is changed from LIST OF BYTE VARYING to BIGINT. This column contains the DBKEY of the original BLOB data. Therefore, the contents of this column should be considered unreliable.
- COMPUTED BY columns in a table are not extracted. These columns are not stored in the after-image journal file.
- VARCHAR fields are not space padded in the output file. The VARCHAR data type is extracted as a 2-byte count field and a fixed-length data field. The 2-byte count field indicates the number of valid characters in the fixed-length data field. Any additional contents in the data field are unpredictable.
- You cannot extract changes to a table when the table definition is changed within an after-image journal file. Data definition language (DDL) changes to a table are not allowed within an .ajj file being extracted. All records in an .ajj file must be the current record version. If you are going to perform DDL operations on tables that you wish to extract using the LogMiner for Rdb, you should:
 1. Back up your after-image journal files.
 2. Extract the .ajj files using the RMU Unload After_Journal command.
 3. Make the DDL changes.
- Do not use the OpenVMS Alpha High Performance Sort/Merge utility (selected by defining the logical name SORTSHR to SYSSSHARE:HYPERSORT) when using LogMiner for Rdb. HYPERSORT supports only a subset of the library sort routines that LogMiner requires. Make sure that the SORTSHR logical name is not defined to HYPERSORT.

8.4 Information Returned by LogMiner for Rdb

LogMiner for Rdb appends several output fields to the data fields, creating an output record. The output record contains fixed-length fields in a binary data format (that is, integer fields are not converted to text strings). The data fields correspond to the extracted table columns. This information may or may not be required by all applications and readers of the data. There is currently no available method to restrict or reorder the output fields.

Extracted data field contents are the fields that are actually stored in the Oracle Rdb database. COMPUTED BY fields are not extracted because they are not stored in the database or in the after-image journal file. Segmented string (BLOB) contents are not extracted.

Table 8–1 describes the output fields and data types of an output record.

Table 8–1 Output Fields

Field Name	Data Type	Description
ACTION	CHAR (1 byte)	Indicates record state. "M" indicates an insert or modify action. "D" indicates a delete action. "E" indicates stream end-of-file (EOF) when a callback routine is being used.
RELATION_NAME	CHAR (31 bytes)	Table name. Space padded to 31 characters.
RECORD_TYPE	LONGWORD INTEGER	The Oracle Rdb internal relation identifier.
DATA_LEN	WORD INTEGER	Length, in bytes, of the data record content.
NBV_LEN	WORD INTEGER	Length, in bits, of the null bit vector content.
DBK	DBKEY (64-bit QUADWORD)	Records logical database key. The database key is a 3-field structure containing a 16-bit line number, a 32-bit page number and a 16-bit area number.
START_TAD	DATE VMS	Date/time of the start of the transaction.
COMMIT_TAD	DATE VMS	Date/time of the commitment of the transaction.
TSN	QUADWORD INTEGER	Transaction sequence number of the transaction that performed the record operation.
RECORD_VERSION	WORD INTEGER	Record version.
Record Data	Varies	Actual data record field contents.

(continued on next page)

Table 8–1 (Cont.) Output Fields

Field Name	Data Type	Description
Record NBV	BIT VECTOR (array of bits)	Null bit vector. There is one bit for each field in the data record. If a bit value is 1, the corresponding field is NULL; if a bit value is 0, the corresponding field is not NULL and contains an actual data value. The null bit vector begins on a byte boundary. Any extra bits in the final byte of the vector after the final null bit are unused.

8.5 Record Definition Prefix for LogMiner Fields

An RMS file containing the record structure definition for the output file can be used as an input file to the RMU Load command if extracted data is to be loaded into an Oracle Rdb database. The record description uses the CDO record and field definition format (this is the format used by the RMU Load and RMU Unload commands when the Record_Definition qualifier is used). The default file extension is .rrd.

The record definition for the fields that LogMiner for Rdb writes to the output is shown in the following example. These fields can be manually appended to a record definition file for the actual user data fields being unloaded. Alternately, the Record_Definition qualifier can be used with the Table qualifier or within an Options file to automatically create the record definition file. This can be used to load a transaction table within a database. A **transaction table** is the output that LogMiner for Rdb writes to a table consisting of sequential transactions performed in a database.

```

DEFINE FIELD RDB$LM_ACTION          DATATYPE IS TEXT SIZE IS 1.
DEFINE FIELD RDB$LM_RELATION_NAME    DATATYPE IS TEXT SIZE IS 31.
DEFINE FIELD RDB$LM_RECORD_TYPE      DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD RDB$LM_DATA_LEN         DATATYPE IS SIGNED WORD.
DEFINE FIELD RDB$LM_NBV_LEN          DATATYPE IS SIGNED WORD.
DEFINE FIELD RDB$LM_DBK              DATATYPE IS SIGNED QUADWORD.
DEFINE FIELD RDB$LM_START_TAD        DATATYPE IS DATE
DEFINE FIELD RDB$LM_COMMIT_TAD       DATATYPE IS DATE
DEFINE FIELD RDB$LM_TSN              DATATYPE IS SIGNED QUADWORD.
DEFINE FIELD RDB$LM_RECORD_VERSION   DATATYPE IS SIGNED WORD.

```

8.6 SQL Table Definition Prefix for LogMiner Fields

The SQL record definition for the fields that LogMiner for Rdb writes to the output is shown in the following example. These fields can be manually appended to the table creation command for the actual user data fields being unloaded. Alternately, the Table_Definition qualifier can be used with the Table qualifier or within an Options file to automatically create the SQL definition file. This can be used to create a transaction table of changed data.

```

SQL> create table MYLOGTABLE (
cont> RDB$LM_ACTION          CHAR,
cont> RDB$LM_RELATION_NAME   CHAR (31),
cont> RDB$LM_RECORD_TYPE     INTEGER,
cont> RDB$LM_DATA_LEN        SMALLINT,
cont> RDB$LM_NBV_LEN         SMALLINT,
cont> RDB$LM_DBK             BIGINT,
cont> RDB$LM_START_TAD       DATE VMS,
cont> RDB$LM_COMMIT_TAD     DATE VMS,
cont> RDB$LM_TSN             BIGINT,
cont> RDB$LM_RECORD_VERSION  SMALLINT ...);

```

8.7 Segmented String Columns

Segmented string (also called BLOB or LIST OF BYTE VARYING) column data is not extracted. However, the field definition itself is extracted as a quadword integer representing the database key of the original segmented string data. In generated table definition or record definition files, a comment is added indicating that the segmented string data type is not supported by LogMiner for Rdb.

8.8 Additional Examples

The following sections contain additional examples.

8.8.1 Example .rrd for the EMPLOYEES Table

The following example is the transaction table record definition (.rrd) file for the EMPLOYEES table from the PERSONNEL database:

```

DEFINE FIELD RDB$LM_ACTION          DATATYPE IS TEXT SIZE IS 1.
DEFINE FIELD RDB$LM_RELATION_NAME   DATATYPE IS TEXT SIZE IS 31.
DEFINE FIELD RDB$LM_RECORD_TYPE     DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD RDB$LM_DATA_LEN        DATATYPE IS SIGNED WORD.
DEFINE FIELD RDB$LM_NBV_LEN         DATATYPE IS SIGNED WORD.
DEFINE FIELD RDB$LM_DBK             DATATYPE IS SIGNED QUADWORD.
DEFINE FIELD RDB$LM_START_TAD       DATATYPE IS DATE.
DEFINE FIELD RDB$LM_COMMIT_TAD     DATATYPE IS DATE.
DEFINE FIELD RDB$LM_TSN             DATATYPE IS SIGNED QUADWORD.
DEFINE FIELD RDB$LM_RECORD_VERSION  DATATYPE IS SIGNED WORD.

DEFINE FIELD EMPLOYEE_ID            DATATYPE IS TEXT SIZE IS 5.
DEFINE FIELD LAST_NAME              DATATYPE IS TEXT SIZE IS 14.
DEFINE FIELD FIRST_NAME             DATATYPE IS TEXT SIZE IS 10.
DEFINE FIELD MIDDLE_INITIAL         DATATYPE IS TEXT SIZE IS 1.
DEFINE FIELD ADDRESS_DATA_1        DATATYPE IS TEXT SIZE IS 25.
DEFINE FIELD ADDRESS_DATA_2        DATATYPE IS TEXT SIZE IS 20.
DEFINE FIELD CITY                   DATATYPE IS TEXT SIZE IS 20.
DEFINE FIELD STATE                  DATATYPE IS TEXT SIZE IS 2.
DEFINE FIELD POSTAL_CODE            DATATYPE IS TEXT SIZE IS 5.
DEFINE FIELD SEX                    DATATYPE IS TEXT SIZE IS 1.
DEFINE FIELD BIRTHDAY               DATATYPE IS DATE.
DEFINE FIELD STATUS_CODE            DATATYPE IS TEXT SIZE IS 1.

```

```

DEFINE RECORD EMPLOYEES.
  RDB$LM_ACTION .
  RDB$LM_RELATION_NAME .
  RDB$LM_RECORD_TYPE .
  RDB$LM_DATA_LEN .
  RDB$LM_NBV_LEN .
  RDB$LM_DBK .
  RDB$LM_START_TAD .
  RDB$LM_COMMIT_TAD .
  RDB$LM_TSN .
  RDB$LM_RECORD_VERSION .
  EMPLOYEE_ID .
  LAST_NAME .
  FIRST_NAME .
  MIDDLE_INITIAL .
  ADDRESS_DATA_1 .
  ADDRESS_DATA_2 .
  CITY .
  STATE .
  POSTAL_CODE .
  SEX .
  BIRTHDAY .
  STATUS_CODE .
END EMPLOYEES RECORD.

```

8.8.2 Callback Module for the EMPLOYEES Table

The following C source code segment demonstrates the structure of a module that can be used as a callback module and routine to process employee transaction information from LogMiner for Rdb. The routine, `Employees_Callback`, would be called by LogMiner for Rdb for each extracted record. Note that the final time the callback routine is called, the `RDB$LM_ACTION` field will be set to "E" to indicate the end of the output stream.

```

#include <stdio>

typedef unsigned char date_type[8];
typedef unsigned char dbkey_type[8];
typedef unsigned char tsn_type[8];

typedef struct {
  unsigned char      rdb$lm_action;
  char               rdb$lm_relation_name[31];
  unsigned int       rdb$lm_record_type;
  unsigned short int rdb$lm_data_len;
  unsigned short int rdb$lm_nbv_len;
  dbkey_type         rdb$lm_dbk;
  date_type          rdb$lm_start_tad;
  date_type          rdb$lm_commit_tad;
  tsn_type           rdb$lm_tsn;
  unsigned short int rdb$lm_record_version;
  char               employee_id[5];
  char               last_name[14];
  char               first_name[10];
  char               middle_initial[1];
  char               address_data_1[25];
  char               address_data_2[20];
  char               city[20];
  char               state[2];
  char               postal_code[5];
  char               sex[1];
  date_type          birthday;
  char               status_code[1];
} transaction_data;

```

```

void employees_callback (unsigned int data_len, transaction_data data_buf)
{
    .
    .
    .
    return;}

```

Use the C compiler (either VAX C or DEC C) to compile this module. When linking this module, the symbol EMPLOYEES_CALLBACK needs to be externalized in the sharable image. Refer to the OpenVMS manual discussing the Linker utility for more information about creating sharable images.

On OpenVMS Alpha systems, you can use a LINK command similar to the following:

```

$ LINK /SHARABLE = EXAMPLE.EXE EXAMPLE.OBJ + SYS$INPUT: /OPTIONS
SYMBOL_VECTOR = (EMPLOYEES_CALLBACK = PROCEDURE)
<Ctrl/Z>

```

On OpenVMS VAX systems, you can use a LINK command similar to the following:

```

$ LINK /SHARABLE = EXAMPLE.EXE EXAMPLE.OBJ + SYS$INPUT: /OPTIONS
UNIVERSAL = EMPLOYEES_CALLBACK
<Ctrl/Z>

```

8.8.3 Using LogMiner and the RMU Load Command to Replicate Table Data

You can use triggers and a transaction table to construct a method to replicate table data from one database to another using RMU Unload After_Journal and RMU Load commands based on transactional changes to the source table. This data replication method requires no programming. Instead, existing features of Oracle Rdb can be combined to provide this functionality.

For this example, consider a simple customer information table called CUST with a unique customer ID value, customer name, address, and postal code. Changes to this table are to be moved from an OLTP database to a reporting database system on a periodic (perhaps nightly) basis.

First, in the reporting database, a customer table of the same structure as the OLTP customer table is created. In this example, this table is called RPT_CUST. It contains the same fields as the OLTP customer table called CUST.

```

SQL> CREATE TABLE RPT_CUST
      CUST_ID          INTEGER,
      CUST_NAME        CHAR (50),
      CUST_ADDRESS     CHAR (50),
      CUST_POSTAL_CODE INTEGER;

```

Next, a temporary table is created in the reporting database for the LogMiner extracted transaction data from the CUST table. This temporary table definition specifies ON COMMIT DELETE ROWS so that data in the temporary table is deleted from memory at each transaction commit. A temporary table is used because there is no need to journal changes to the table.


```

SQL> CREATE GLOBAL TEMPORARY TABLE RDB_LM_RPT_CUST (
  RDB$LM_ACTION          CHAR,
  RDB$LM_RELATION_NAME  CHAR (31),
  RDB$LM_RECORD_TYPE    INTEGER,
  RDB$LM_DATA_LEN       SMALLINT,
  RDB$LM_NBV_LEN        SMALLINT,
  RDB$LM_DBK            BIGINT,
  RDB$LM_START_TAD      DATE VMS,
  RDB$LM_COMMIT_TAD     DATE VMS,
  RDB$LM_TSN            BIGINT,
  RDB$LM_RECORD_VERSION SMALLINT,
  CUST_ID               INTEGER,
  CUST_NAME             CHAR (50),
  CUST_ADDRESS          CHAR (50),
  CUST_POSTAL_CODE      INTEGER) ON COMMIT DELETE ROWS;

```

For data to be populated in the RPT_CUST table in the reporting database, a trigger is created for the RDB_LM_RPT_CUST transaction table. This trigger is used to insert, update, or delete rows in the RPT_CUST table based on the transaction information from the OLTP database for the CUST table. The unique CUST_ID field is used to determine if customer records are to be modified or added.

```

SQL> CREATE TRIGGER RDB_LM_RPT_CUST_TRIG
cont> AFTER INSERT ON RDB_LM_RPT_CUST
cont>
cont> -- Modify an existing customer record
cont>
cont> WHEN (RDB$LM_ACTION = 'M' AND
cont>       EXISTS (SELECT RPT_CUST.CUST_ID FROM RPT_CUST
cont>               WHERE RPT_CUST.CUST_ID = RDB_LM_RPT_CUST.CUST_ID))
cont>   (UPDATE RPT_CUST SET
cont>       RPT_CUST.CUST_NAME = RDB_LM_RPT_CUST.CUST_NAME,
cont>       RPT_CUST.CUST_ADDRESS = RDB_LM_RPT_CUST.CUST_ADDRESS,
cont>       RPT_CUST.CUST_POSTAL_CODE = RDB_LM_RPT_CUST.CUST_POSTAL_CODE
cont>       WHERE RPT_CUST.CUST_ID = RDB_LM_RPT_CUST.CUST_ID)
cont> FOR EACH ROW
cont>
cont> -- Add a new customer record
cont>
cont> WHEN (RDB$LM_ACTION = 'M' AND NOT
cont>       EXISTS (SELECT RPT_CUST.CUST_ID FROM RPT_CUST
cont>               WHERE RPT_CUST.CUST_ID = RDB_LM_RPT_CUST.CUST_ID))
cont>   (INSERT INTO RPT_CUST VALUES
cont>       (RDB_LM_RPT_CUST.CUST_ID,
cont>       RDB_LM_RPT_CUST.CUST_NAME,
cont>       RDB_LM_RPT_CUST.CUST_ADDRESS,
cont>       RDB_LM_RPT_CUST.CUST_POSTAL_CODE))
cont> FOR EACH ROW
cont>
cont> -- Delete an existing customer record
cont>
cont> WHEN (RDB$LM_ACTION = 'D')
cont>   (DELETE FROM RPT_CUST
cont>     WHERE RPT_CUST.CUST_ID = RDB_LM_RPT_CUST.CUST_ID)
cont> FOR EACH ROW;

```

Within the trigger, the action to take (for example, to add, update, or delete a customer record) is based on the RDB\$LM_ACTION field (which will be defined as D or M) and the existence of the customer record in the reporting database. For modifications, if the customer record does not exist, it is added; if it does exist, it is updated. For a deletion on the OLTP database, the customer record is deleted from the reporting database.

The RMU Load command is used to read the output from LogMiner for Rdb and load the data into the temporary table where each insert will result in the trigger executing. The Commit_Every qualifier is used to avoid filling memory with the customer records in the temporary table because as soon as the trigger executes, the record in the temporary table is no longer needed.

```
$ RMU /UNLOAD /AFTER_JOURNAL OLTP.RDB OLTP.AIJBCK -  
      /TABLE = (NAME = CUST,  
              OUTPUT = CUST.DAT,  
              RECORD_DEFINITION = RDB_LM_RPT_CUST.RRD)  
  
$ RMU /LOAD REPORT_DATABASE.RDB RDB_LM_RPT_CUST CUST.DAT -  
      /RECORD_DEFINITION = FILE = RDB_LM_RPT_CUST.RRD -  
      /COMMIT_EVERY = 1000
```

8.8.4 Using LogMiner to Minimize Application Downtime for Maintenance

Lengthy offline application or database maintenance operations can pose a significant problem in high-availability production environments. The LogMiner for Rdb feature can help reduce the length of downtime to a matter of minutes.

If a back up of the database is used for maintenance operations, the application can continue to be modified during lengthy maintenance operations. Once the maintenance is complete, the application can be shut down, the production system .aj file or files can be backed up, and LogMiner for Rdb can be used to extract changes made to production tables since the database was backed up. These changes can then be applied (using an application program or the trigger technique previously described) to the new database. Once the new database has been updated, the application can be restarted using the new database.

The sequence of events required would be similar to the following:

1. Perform a full online, quiet-point database backup of the production database.
2. Restore the backup to create a new database that will eventually become the production database.
3. Perform maintenance operations on the new database. (Note that the production system continues to run.)
4. Perform an online, quiet-point after-image journal backup of the production database.
5. Use the RMU Unload After_Journal command to unload all database tables into individual output files from the .aj backup file.
6. Using either the trigger technique or an application program, update the tables in the new database with the changed data.
7. Shut down the production application and close the database.
8. Perform an offline, quiet-point after-image journal backup of the production database.
9. Use the RMU Unload After_Journal command to unload all database tables into individual output files from the .aj backup file.
10. Using either the trigger technique or an application program, update the tables in the new database with the changed data.
11. Start an online, quiet-point backup of the new database.
12. Change logical names or the environment to specify the new database root file as the production database.

13. Restart the application on the new database.

Depending on the amount of application database activity, steps 4, 5, and 6 can be repeated to limit the amount of data that needs to be applied (and the amount of downtime required) during the final after-image journal backup and apply stage in steps 8, 9, and 10.

8.8.5 Using an OpenVMS Pipe

You can use an OpenVMS pipe to pass data from the RMU Unload After_Journal command to another application (for example, RMU Load). Do not use any options (such as the Log or Verify qualifiers) that could cause LogMiner to send extra output to the SYS\$OUTPUT device, as that information would be part of the input data source stream to the next pipeline segment.

You may find that the OpenVMS default size of the pipe is too small if the records being extracted (including LogMiner fields) are larger than 256 bytes. If the pipe is too small, increase the SYSGEN parameters MAXBUF and DEFMBXMXMSG, and then reboot the system.

The following example uses LogMiner for Rdb to direct output to an OpenVMS pipe device and uses RMU Load to read the pipe device as the input data record stream. Using the pipeline allows parallel processing and also avoids the need for an intermediate disk file. Note that you must have created the record definition (.rrd) file prior to executing the command.

```
$ PIPE (RMU /UNLOAD /AFTER_JOURNAL OLTP.RDB AIJ1.AIJ -  
  /TABLE = (NAME = MYTBL, OUTPUT = SYS$OUTPUT:)) -  
  | (RMU /LOAD REPORTS.RDB MYLOGTBL SYS$PIPE: -  
    /RECORD_DEFINITION = FILE = MYLOGTBL.RRD)
```

Implementing Row Cache

A.1 Overview

A.1.1 Introduction

Oracle Rdb uses buffers to temporarily store database pages during read and update operations. When you create or modify a database, you can set up buffers for database pages in either of the following ways:

- Local Buffers

Database users have their own set of private local database page buffers. Data of interest is read from disk into a local database page buffer. Local buffers are not shared among users. Sharing occurs only when a database page is written back to disk and another user retrieves that database page. The sharing is done at the physical page level and can be I/O intensive.

- Global Buffers

Database users on the same system share a common set of global database page buffers that reside in global memory. Database pages that are read from disk by one user can be seen directly by another user. Little or no I/O is needed to share global buffers; however, sharing data is still done at the level of database page buffers. A database page buffer has a fixed size across all storage areas in the database. The amount of data in a database page buffer that is of interest to multiple users may be small compared to its overall size. Although this model may be more efficient than using local buffers, there are better ways to share data among users.

Oracle Rdb offers a feature called row caching to enhance the performance of memory buffers. Because row caching is a cache of rows, you can use it in conjunction with local or global database page buffers. Please consider, however, that when using both global buffers and row cache, you could have two copies of data consuming your global memory—one copy in the row cache and one in a global buffer. Note also that row caches are not designed to be an “in-memory database”. As its name implies, a **row cache** is a set of database rows that reside in memory between the users and the rest of the database rows on disk. Data rows, system records, as well as hashed and sorted index nodes, can be cached. Access to a row in a row cache is through its logical database key (dbkey).

All processes attached to a database share a pool of row occurrences that reside in shared memory row caches. No disk I/O is needed to share a row in a row cache. Only the rows of interest, not the physical pages, are kept in shared memory, thereby increasing the use of shared memory. In addition, you can create many row caches, each with its own row size. Row caches can be used to efficiently store rows of specific sizes from specified tables. The Oracle Rdb implementation of row caches gives you the option to specify portions of row caches to occupy process private virtual memory, shared global pagefile sections on OpenVMS systems, or shared physical main memory. Oracle Rdb row caching also allows

you to use very large memory (VLM) on OpenVMS Alpha systems. Subsequent sections provide more detail on each of these options.

The row caching feature is designed to improve performance through reduced I/O operations by finding rows of interest in the row cache instead of accessing them on disk. The greater number of times the data is located in the row cache, the more useful the cache is and better overall performance results.

The next section describes how row caching works with basic Oracle Rdb database functions.

A.1.2 Database Functions Using Row Cache

The following list describes how common database operations use the row caching feature.

- **Fetching Data**

When you request a row from a database, Oracle Rdb first checks to see if the requested row is located in a row cache. If the row is in a row cache, the row is retrieved from the cache. If the row is not in a cache, Oracle Rdb checks the page buffer pool. If the row is not in the page buffer pool, Oracle Rdb performs a disk I/O operation to retrieve the row. The requested row is then inserted into the row cache, if possible.
- **Storing Data**

When a new row is stored in the database, Oracle Rdb may perform a disk I/O operation to find space for the new row and get a dbkey for the row. Once space has been reserved on a database page, Oracle Rdb checks for a row cache in which to put the new row. The new row is inserted into a row cache, if possible.
- **Modifying Data**

If a modification to a row in a cache causes the row to grow (replaces a null value, for example), then the database page must be modified to reserve additional space for that row. If the database page does not have room for the modified row, resulting in fragmentation, then the row is deleted from the cache. If the modification keeps the row the same size or makes it smaller, then the modified row remains in the cache and no database page is accessed. This means that the unused space on the page is not reclaimed and hence is not immediately available for reuse. Compressed rows and indexes that are modified are more likely to require database access than uncompressed ones.
- **Deleting Data**

If the row is in a row cache, Oracle Rdb sets the length of the row to zero to erase it. It is not erased from the database page on disk immediately. Therefore, the deleted space is not reusable immediately.
- **When snapshots are enabled**

During a read-only transaction, Oracle Rdb first checks to see if the row is in a row cache. If the row is found and is visible to the transaction, the row is returned from the row cache and no disk I/O operation is necessary. If the row is not visible, Oracle Rdb must find the visible version of this row in the snapshot file. Information stored in the row cache, however, can shorten the search and thereby reduce I/O operations to the snapshot file.

During a read/write transaction that is performing an update, Oracle Rdb writes the before-image of the data to the snapshot file. Oracle Rdb writes the before-image information out to the snapshot file each time a row in the user's row cache **working set** is modified. If a row falls out of the working set list and is remodified later in the transaction, the before-image information is written back to the snapshot file when the row re-enters the working set.

Global and local buffers use the least-recently used (LRU) replacement strategy for database pages. Row caching uses a modified form of the LRU replacement strategy. Each database user can protect the last 10 rows they accessed. This group of rows is referred to as a **working set**. Rows that belong to a working set are considered to be **referenced** and are not eligible for row replacement.

During a read/write transaction that performs a delete operation, the processing is the same as described in the previous paragraphs.

A.1.3 Writing Modified Rows to Disk

With row caching, many data modifications are performed on the in-memory copy of the data. Therefore, Oracle Rdb must have a way to write these rows to storage on disk.

The following list describes the ways that modified rows can be written back to the database page on disk.

- If the page on which a modified row resides is in the user's buffer pool and is already locked by the user when the update to that row must be recorded in the row cache, then the update is made to the row in the cache and on the database page.

In this case, the row cache entry is not considered to be marked or modified. This situation occurs when a transaction is committed or when a row is flushed from a row cache.

- During an undo operation, the before-image of each modified row is placed on the database page.

An **undo** operation occurs as part of an aborted SQL statement, transaction rollback, or database recovery of a terminated user's process.

- During a **redo** operation, the after-image of each modified row is stored on the database page only if recovering from a node failure. If recovering from a process failure, no redo is done for in-memory row cache modifications because the row cache memory is still valid and intact. (Changes made to database pages are still redone.)
- During a row cache checkpoint operation, all modified rows (or all rows) from the row caches are written to disk storage.

This is the most common method of writing updated rows back to disk storage.

- During a row cache sweep operation, a set of modified rows are written back to the database from the row cache. After the rows are written back to disk, the space they occupied is considered selectable for reuse.

A row cache sweep operation is initiated when a user process tries to insert rows into a row cache and finds no free space available.

A.1.4 Row Cache Checkpointing and Sweeping

Checkpointing and sweeping operations are critical in performing the operations necessary to write modified, committed rows back to disk from a row cache. The row cache server (RCS) process performs these tasks. There is one RCS process per database. Any failure of the RCS process forces the shutdown of the entire database.

To monitor the status of rows in a row cache, Oracle Rdb maintains a modification flag for every row in a cache to indicate which rows have been modified. The modification flags are shown in the following table:

Modification Flag	Meaning
Marked	The row has been modified in the row cache only. If this modification remains only in the row cache at the time the transaction is committed, then this marked flag indicates this row in the row cache is not reflected in the database.
Hot	The marked row has been modified since the last checkpoint.
Cold	The marked row has not been modified since the last checkpoint.

The RCS process performs three types of operations:

- Synchronous operations where the requester is waiting for the operation to complete

The following are operations of this type:

- The RCS process checkpoint operation that is part of an AIJ fast-commit checkpoint

For example, if the RMU Checkpoint command with the Wait qualifier is issued, then the requester will wait for the RCS process to complete its checkpoint.

- A checkpoint to the database for all row caches before certain database utility operations can begin
- Row cache checkpoint operations

Checkpointing is a repetitive, time-driven event that writes rows from all row caches back to disk storage. The RCS process writes data to a cache backing file (.rdc) or directly to the database for each cache, depending on how the row cache was defined. The time interval at which a checkpoint occurs is also programmable. When the last user detaches from the database, the RCS process performs a final checkpoint operation to the database (never to the cache backing files). See Section A.4.2.1 for more details.

- Row cache sweep operations

Sweeping is done to make space available in a particular row cache. When a transaction requests space and none is available, the RCS process sweeps marked rows back from the particular row cache to the database. It also resets row cache reference counts if your database has experienced some user process failures. This creates free memory for subsequent transactions to insert rows into each cache. This may never be necessary if checkpointing is done at appropriate intervals. See Section A.4.2.3 for more details.

The RCS process selects work requests based on their priority; synchronous operations are checked first, then checkpoints, followed by sweep operations.

If a database is opened manually, the RCS process is started as part of the open operation. If a database is opened automatically, the RCS, by default, is started when a row cache is referenced for the first time.

When the last user disconnects from the database (with the database open setting set to automatic) or when the database is closed manually, the RCS process performs a final checkpoint to the database. When this operation completes, all marked rows have been written back to the database. The RCS process writes out its checkpoint information to indicate that backing files are no longer needed if there is a need to recover from a node failure. At this time, the cache backing files, if any, are deleted by default. If you want to preserve the backing files and have them be reused at database startup, define the logical `RDM$BIND_RCS_KEEP_BACKING_FILES` to "1".

Details of the RCS actions can be seen by creating an RCS process log file. Before opening the database, define the `RDM$BIND_RCS_LOG_FILE` system logical name to indicate the device, directory, and file name of the RCS process log file you want to create. If no device and directory are specified, the RCS log file is created in the same directory as that which contains the database root file.

A.1.5 Node and Process Failure Recovery

The following sections describe how the row cache feature interacts with node and process failure recovery.

To understand how database recovery works with row caches, you should understand the interactions that occur when writing to row caches, writing to the recovery-unit journal (RUJ) files, and writing to the after-image journal (AIJ) files. This interaction is identical to the interactions that occur among database page buffers, RUJ journaling, and AIJ journaling. For more information, see the *Oracle Rdb Guide to Database Performance and Tuning*.

The AIJ fast commit feature is a prerequisite for enabling row caching. This means that updates to the database are not flushed back to the database pages at the time a transaction is committed. In the case of row caching, the modified rows reside in the in-memory row caches. However, all after-image (updated rows) must be flushed to the AIJ file when the transaction is committed. In the event of a failure, the committed, updated rows can be reapplied to the database from the AIJ file.

Recovery-unit journaling is critical in ensuring that rows can be returned to their previous state when either a SQL statement or transaction rolls back or aborts abnormally. A row's before-image must be preserved BEFORE any modification is made to a row on a database page or in a row cache. Before-images are placed in an in-memory RUJ buffer. Only when that buffer becomes full or when a modified page or modified row cache entry is being put back must the RUJ information first be synchronously written to the RUJ file. For a database without row caches, this means the write IO to the RUJ file must be performed before a database page containing a modified row can be written to disk.

With row caches, Oracle Rdb is frequently modifying only memory, not database pages. The requirement for RUJ information being written BEFORE a modification is put back into the row cache still exists. Writing synchronous IOs to the RUJ before modifying in-memory row caches doesn't make much sense. Oracle Rdb minimizes this behavior in two ways:

- A modification to a row cache entry is first done in a local copy. Only when this local copy of the row must be flushed back to the row cache is the RUJ information written out.

- The RUJ buffer resides in a system-wide, shared memory global section that is visible to the DBR process. Therefore the before-image rows don't have to be written to the RUJ file unless an uncommitted modification to a database page (a store or a modify bigger operation) is forced to disk or when the RUJ buffer overflows.

The global section created for the RUJ buffers will be about 256 VAX pages or 16 Alpha pages for each allowed user of a database. One global section is created for each database that has row caching enabled. To disable this optimization for databases with row caching enabled, define the logical name RDM\$BIND_RUJ_GLOBAL_SECTION_ENABLED to "0" in the system logical name table.

You need to increase several OpenVMS system parameters, as follows:

- **GBLSECTIONS**
Increase by the maximum number of Oracle Rdb databases open at one time on the system.
- **GBLPAGES**
Increase by 256 times the maximum number of users for each database open at one time on the system.
- **GBLPAGFIL**
Increase by 256 (on OpenVMS VAX systems) or by 16 (on OpenVMS Alpha systems), times the maximum number of users for each database open at one time on the system.

There is no additional virtual memory consumption for database users when the RUJ global buffers optimization is enabled; each user process continues to use the same amount of virtual memory (256 blocks) as when the optimization is not enabled.

Databases that do not have row caching enabled will not have optimization enabled for the RUJ buffer in a global section.

A.1.5.1 Process Failure

When a process terminates abnormally, Oracle Rdb activates a database recovery (DBR) process to recover the work done by the terminated user. The DBR process first performs transaction REDO, reapplying committed transactions' modifications to the database pages that had only been written to the AIJ file back to the database. Because the row cache memory is still in tact, in-memory row cache changes do not have to be redone during REDO. The DBR process then proceeds to UNDO the user's outstanding transaction. If the RUJ system-wide process buffers are enabled, the DBR process first writes the current RUJ buffer to the RUJ file. It then recovers the RUJ file by placing the before-image of each row back on the database page. If the dbkey for that row is also found in a row cache, the before-image is placed back into the row cache too.

A.1.5.2 Node Failure

There are several events that constitute node failure to Oracle Rdb:

- Machine or operating system fails
- The Oracle Rdb monitor process terminates unexpectedly
- The Oracle Rdb RCS process terminates unexpectedly
- An Oracle Rdb DBR process terminates unexpectedly
- The RMU Monitor Stop command is issued with the Abort=delprc qualifier

- The RMU Close command is issued with the Abort=delprc qualifier

All of these events cause all access to an Oracle Rdb database to cease immediately. Recovery from a node failure event is deferred until the next time the database is attached or opened. Even if the RMU Open command with the Row_Cache=disabled qualifier is executed next, this will initiate recovery from the node failure. It will not create nor populate the in-memory row caches during the recovery. Once recovery has finished, no row caches will be active while the database stays open in this manner.

Oracle Rdb has several schemes for recovering a database after a node failure. For a database without row caching enabled and without global buffers enabled, Oracle Rdb recovers from a node failure by creating one DBR process for each abnormally terminated user and these DBR processes recover the database in parallel. For a database without row caching enabled but with global buffers enabled, Oracle Rdb recovers one database user at a time by creating one DBR process at a time. For a database with row caching enabled, Oracle Rdb creates one DBR process and that process performs recovery for all the users.

For recovery from a node failure for a database with row caching enabled, the DBR process performs recovery in the following steps.

1. Recovers the backing files. For each row cache that is checkpointed to a backing file, the DBR process:
 - Reads each row from the backing file.
 - If the row has been updated (marked), then the DBR process writes this row back to the appropriate database page.
 - Inserts this row into the empty row cache in shared memory. If the database is opened with row caching disabled or if the system logical name RDMSBIND_DBR_UPDATE_RCACHE is defined to "0", then the row caches are not repopulated from the backing files.
 - Places this dbkey in a row cache dbkey list.
2. Performs a REDO operation from the oldest user checkpoint. This includes the RCS process checkpoint when the RCS process last checkpointed the row caches.
 - For each transaction rolled back, the DBR process discards the updates.
 - For each transaction committed, the DBR process reapplies those updates to the database pages.

Please note that ALL committed transactions since the oldest checkpoint are applied, not just all committed transactions for the users who were active at the time of the node failure.

 - If DBR is re-populating the row caches and this dbkey is found in the row cache dbkey list, then this occurrence replaces the current one in the row cache. If a row in a mixed format area is erased, it is removed from the row cache and its dbkey is removed from the dbkey list. This is necessary to prevent the physical dbkey that may be reused for a different table or index from being placed in the prior occurrence's row cache.
 - Once the redo operation is completed, the DBR process updates all users' checkpoints to be the current AIJ end-of-file.

3. Performs the UNDO operation for each aborted user's incomplete transaction, if any. The DBR process reads the before-images from the user's RUJ file and writes them back to the database. If the dbkey also exists in a row cache, then the before-image is also written to its row cache entry.

A.1.5.3 The RCS Process and Database Recovery

Because the RCS process and the DBR process both access the row cache structures, they must coordinate their activities. When a DBR process is activated, it immediately notifies the RCS process of its existence using a lock. Then the RCS process aborts whatever request it is performing, requeues the request at the head of the appropriate queue, and waits for the database recovery activity to complete. Upon completion of database recovery, the RCS process resumes its operations by executing the next operation based on priority.

A.1.6 Considerations When Using the Row Cache Feature

This section contains further information on using the row cache feature.

- Hot Standby

Row caching is not allowed to be active on the standby database. Because the AIJ file does not contain logical dbkeys, there is no way to maintain rows in the cache on the standby system. On the standby system, issue the RMU Open command with the Row_Cache=Disabled qualifier to open the database without activating row caching. If failover is necessary, simply close the standby database and reopen it normally. Your standby database will have row caches activated.

- Backing files

If you are using row cache backing files, then do not use Hot Standby on the same machine as the master database. Both databases will attempt to use the same backing files.

Similarly, do not attempt to use the same directory location for backing files for two or more databases if any of their row cache names are identical. Multiple databases will attempt to use the same backing files.

- Utilities that access the database pages directly

Some RMU commands do not access data by logical dbkey but instead read the database pages directly. These commands cannot access the row caches directly. Oracle Rdb resolves this problem by having each command request the RCS process write all marked rows back to the database. The RMU operation waits for this task to complete.

The RMU commands affected by this are:

- Backup online
- Analyze
- Verify
- Copy database online

These operations may exhibit a delay in starting. If you specify the RMU log qualifier, Oracle Rdb will output a message when it is waiting for the RCS request and when the RCS request has completed. If your database's row caches are set to checkpoint to the database rather than to backing files, then this delay will be minimized.

- Sequential scans

When the execution strategy for a query is a sequential scan, Oracle Rdb scans the physical areas by performing the same I/O operations it would do if there were not any row caches. The major reasons for this are as follows:

- Oracle Rdb does not have a list of all dbkeys in an area; it materializes them by reading all pages and examining all lines on each page. However, data is returned from the row cache if it is found there. Although Oracle Rdb reads the database pages to find the dbkeys of rows in the table, it still needs to look in the cache to see if the row is there. A row in the cache contains more recent data than that which is on disk.
- There is no guarantee that all rows in a sequential scan can fit in a row cache. Row caches are often sized to include a percentage of the total number of rows where the most commonly used rows can be shared in memory.

Oracle Rdb is designed to avoid populating the cache during a strict sequential scan. It is designed this way because otherwise a query performing a sequential scan of a table looking for just a few records would fill the cache with every record and might force existing data in the cache back to disk. This would result in a row cache filled with records that you do not need in the cache.

However, note that a sequential *index* scan will populate the cache with data, index rows, or both.

- Snapshots enabled

The Oracle Rdb snapshot mechanism of preserving a consistent view of the database for read-only transactions is not changed by the row cache feature. The before-images of rows needed by read-only transactions are preserved when read/write transactions write them to the snapshot files. Therefore, when snapshots are enabled, update operations are written to the rows in the row cache and the before-image of the row is written to disk. Oracle Rdb has optimized the snapshot mechanism with row caches, however, so that the performance of readers and writers may be better with row caches than without.

The performance of row caches is typically much faster when snapshots are disabled. All of the disk I/O operations necessary to read and write to the snapshot file are eliminated. This is the ideal situation.

- Fragmented rows

Fragmented rows are not stored in the row cache. They are created by fetching the fragments from the database and materializing them in process-private virtual memory.

- Vertical record partitioning

When a logical cache is defined for a vertically partitioned table, each partition of a row is cached as a separate row cache entry. Only partitions that your query references and that can fit are inserted into the row cache.

- Unexpected storage area growth

Oracle Rdb has optimized row caching to minimize the disk I/O operations required. Frequently operations are performed in-memory only. Having the faster performance of in-memory updates is beneficial. However, when you make modifications that keep a row at its current size or smaller, or you make deletions, the database page does not reflect the amount of space that is in use. Even though the row is logically smaller or erased from the database,

it has not been physically removed from the database page. The space it occupies cannot be reused by another transaction until this row is finally written back to the database, usually by the RCS process during a sweep or checkpoint operation, depending on your row cache settings. Because of this, storage areas may grow larger than anticipated. If space reclamation is critical for some storage areas, then consider checkpointing their row caches to the database on a regular basis.

A.2 Requirements for Using Row Caches

To use the row cache feature, an Oracle Rdb database must meet the following configuration requirements:

- The number of cluster nodes must be one.
- After-image journaling must be enabled.
- Fast commit must be enabled.
- One or more row cache slots must be reserved.
- Row caching must be enabled.

Use the RMU Dump command with the Header qualifier to see if you have met the requirements for using row caches. In the following example, warnings are displayed for row cache requirements that have not been met.

```
$ RMU/DUMP/HEADER INVENTORY
.
.
.
Row Caches...
- Active row cache count is 4
- Reserved row cache count is 20
- Checkpoint information
  Time interval is 10 seconds
  Default source is updated rows
  Default target is backing file
  Default backing file directory is "DISK1:[CACHE]"
- WARNING: Maximum node count is 16 instead of 1
- WARNING: After-image journaling is disabled
- WARNING: Fast commit is disabled
.
.
.
```

A.3 Designing and Creating a Row Cache

The following sections describe considerations for designing and creating row caches.

A.3.1 Reserving Slots for Row Caches

When you create a database, reserve enough row cache slots for both current and future needs. To reserve additional slots and to add or drop a row cache, the database must be closed.

Use the `RESERVE n CACHE SLOTS` clause of the `CREATE DATABASE` or `ALTER DATABASE` statement to reserve slots for row caches, as shown in the following example:

```
SQL> CREATE DATABASE FILENAME INVENTORY
      .
      .
      .
cont> RESERVE 20 CACHE SLOTS;
```

If you do not specify a `RESERVE n CACHE SLOTS` clause, Oracle Rdb reserves one slot by default.

A.3.2 Row Cache Types

The two types of row caches are described in the following list:

- **Physical area**
 You can create a general row cache that is shared by all row types that reside in one or more storage areas. This is the basic type of row cache, called a **physical area row cache**. Because physical area row caches are defined for a storage area, multiple storage areas can map to the same physical area row cache. A physical area row cache can contain all row types in a storage area. In addition, when a physical area row cache is defined, all rows of different sizes in the specified storage area are candidates for the row cache.
 See Section A.3.2.1 for an example of how to assign a row cache to a storage area.
- **Logical area**
 You can create logical area row caches when you create a row cache by using the same name as an existing table or index. A **logical area row cache** is associated with all partitions, both horizontal and vertical, of a specific table or index.
 A logical area cache cannot store the system row from a database page in an mixed format area.

You can use both physical and logical caches to store a table and its index.

The following example shows the reason for using different caches for different row types. Assume the following sizes for the rows in a table and hashed index:

- System records of 16 bytes
- Hash buckets of 100 bytes
- Data rows of 320 bytes

If you created one cache for all three row types, with a row size of 320 bytes, much of the allocated memory would be wasted when storing the smaller system record and the hash bucket. Using this method, the amount of memory, excluding overhead, used for one row cache is as follows, assuming 15000 rows in the cache:

```
Total
number   = (# of rows in cache * row length of largest row)
of bytes
          = (15000 * 320)
          = 4800000 bytes
```

It is more efficient to have three caches, one for each of the row types:

- System records of 16 bytes (PARTS_SYS cache)
- Hash buckets of 100 bytes (PARTS_HASH cache)
- Data rows of 320 bytes (PARTS cache)

In this example the system records are stored in a physical cache (PARTS_SYS) while the hash index buckets and data rows are stored in logical caches (PARTS_HASH and PARTS).

The amount of memory, excluding overhead, used with three row caches is computed as follows:

$$\begin{aligned} \text{Total} & \\ \text{number} & = (\# \text{ of rows in cache} * \text{row length of system record}) + \\ \text{of bytes} & \quad (\# \text{ of rows in cache} * \text{row length of hash bucket}) + \\ & \quad (\# \text{ of rows in cache} * \text{row length of data row}) \\ & = (5000 * 16) + \\ & \quad (5000 * 100) + \\ & \quad (5000 * 320) \\ & = 2180000 \text{ bytes} \end{aligned}$$

A.3.2.1 Assigning Storage Areas to Row Caches

When a storage area is associated with a row cache, the row cache can contain all types of rows, if they can fit. This is called a physical area row cache. One storage area can point to one row cache only. Multiple storage areas can be mapped to the same row cache.

You can also define a default row cache for all of the storage areas in the database by using one of the following statements:

- ALTER DATABASE ... ADD STORAGE AREA ... CACHE USING
- ALTER DATABASE .. ALTER STORAGE AREA ... CACHE USING
- CREATE DATABASE ... CREATE STORAGE AREA ... CACHE USING

The following example shows how to assign the same physical row cache to multiple storage areas:

```
SQL> ALTER STORAGE AREA
cont> PART_ID_A_E CACHE USING PARTS_SYS;
SQL> ALTER STORAGE AREA
cont> PART_ID_F_K CACHE USING PARTS_SYS;
```

A.3.2.2 Assigning Tables to Row Caches

A row cache is considered to be a logical area cache if its name is identical to the name of either a table or an index. If a logical area row cache is created for a vertically or horizontally partitioned table or horizontally partitioned index, then all rows in these partitions are mapped to the single logical area row cache. In the following example, a logical area cache called PARTS is created for the PARTS table that is horizontally partitioned across five storage areas:


```

SQL> CREATE STORAGE MAP PARTS_MAP FOR PARTS
cont> --
cont> -- Parts table partitioned by part_id
cont> --
cont> STORE USING (PART_ID)
cont>     IN PART_ID_A_E WITH LIMIT OF ('F')
cont>     IN PART_ID_F_K WITH LIMIT OF ('L')
cont>     IN PART_ID_L_P WITH LIMIT OF ('Q')
cont>     IN PART_ID_Q_U WITH LIMIT OF ('V')
cont>     OTHERWISE IN PART_ID_V_Z
cont>     PLACEMENT VIA INDEX PARTS_HASH;
SQL>
.
.
.
SQL> ALTER DATABASE FILENAME INVENTORY
cont>     ADD CACHE PARTS
cont>     ROW LENGTH IS 100 BYTES
cont>     CACHE SIZE IS 5000 ROWS;

```

Rows from all five partitions of the PARTS table are automatically cached in the PARTS row cache, if they can fit.

A.3.3 Sizing a Row Cache

When you size a row cache, you specify the following:

- Slot Size

The slot size is the fixed length size of each entry in the row cache. This determines the size of the largest row that can be stored in the row cache. Oracle Rdb will not cache a row if it is larger than the cache's slot size. Use the ROW LENGTH IS parameter of the ADD, ALTER, or CREATE CACHE clause to specify the slot size of the row cache.

Oracle Rdb automatically rounds up the row length to the next 4-byte boundary. This is done because longword aligned data structures perform optimally on its supported platforms.

If you do not specify a slot size when creating a logical cache, Oracle Rdb generates a slot size based on the size of the table row or index node. Note, however, that Oracle Rdb finds the nominal row length of tables and indices using the area inventory page (AIP). Under certain circumstances this AIP length may not be the actual length of the row. In addition, some index structures may have no AIP entry at all. If no entry can be found, Oracle Rdb uses a default length of 256 bytes. Also, if the metadata for a table is modified, then the AIP length is not automatically updated. This can result in incorrect cache sizing. See the *Oracle Rdb Guide to Database Performance and Tuning* for more details on AIP lengths.

- Slot count

The slot count is the number of rows that can be stored in the cache. Use the CACHE SIZE IS parameter of the ADD, ALTER, or CREATE CACHE clause to specify the number of rows that can be stored in the cache.

If you do not specify the CACHE SIZE clause, Oracle Rdb creates a cache of 1000 rows by default.

The following example shows a row cache definition:

```
SQL> ADD CACHE PARTS
cont> ROW LENGTH IS 320 BYTES
cont> CACHE SIZE IS 3000 ROWS;
SQL> --
SQL> -- In this example, the slot size is 320 bytes
SQL> -- and the slot count is 3000.
SQL> --
```

It is important to select a proper slot size for the row cache. As stated previously, if a row is too large, Oracle Rdb will not cache the row. This can result in poor system performance because Oracle Rdb always checks the cache for the row before retrieving the row from disk. Use the RMU Dump Area command to determine the sizes of the data rows, hash buckets, and B-tree nodes. Keep in mind that row sizes within a table can vary greatly. If, for example, the largest row stored in a table is 100 bytes, but the majority of the rows range between 40 and 50 bytes, you may not necessarily want to choose 100 bytes for the slot size. However, you should account for most of the rows, including overhead. If you automatically select the largest row size without comparing it to the sizes of the other rows in the table, you might waste memory.

The following example dumps a few pages from the MY_AREA storage area:

```
$ RMU/DUMP/AREA=MY_AREA/START=5/END=10 TEST_DB/OUT=rmu_dump_area.out
```

Search the rmu_dump_area.out file for the occurrences of “total hash bucket” and “static data” as follows:

```
$ SEARCH RMU_DUMP_AREA.OUT "total hash bucket"
.... total hash bucket size: 97
.... total hash bucket size: 118
.... total hash bucket size: 118
.... total hash bucket size: 118
.... total hash bucket size: 118
.... total hash bucket size: 118
.... total hash bucket size: 118
.... total hash bucket size: 118
.... total hash bucket size: 118
.... total hash bucket size: 118
.
.
.
$ SEARCH rmu_dump_area.out "static data"
.... 311 bytes of static data
.... 311 bytes of static data
.... 311 bytes of static data
.... 311 bytes of static data
.... 311 bytes of static data
.... 311 bytes of static data
.... 311 bytes of static data
.... 311 bytes of static data
.... 311 bytes of static data
.... 311 bytes of static data
.... 311 bytes of static data
.... 311 bytes of static data
.
.
.
```

The hash bucket size is 118 bytes and the data row size is 311 bytes. Other rows in this table may require more or less space. It is important to scan a representative sample of random pages to determine the appropriate row size. Oracle Rdb rounds row sizes up to the next longword.

The RMU Show Statistics row caching screens provide performance information on inserting rows into a cache. One of the statistics, “row too big”, indicates that a row is too large to fit into the specified cache. This statistic is also set when a row in a row cache becomes invalid and must be retrieved from the database page. For example, when a row in the row cache grows to the point where it becomes fragmented, it must be removed from the row cache. This is done by “redirecting” this row out of the row cache to disk, by setting its “row too big” attribute. See Section A.5.1 for more information on the RMU Show Statistics screens related to row caching.

The slot count multiplied by the slot size specifies the approximate size, in bytes, of the row cache. You should also take into account additional overhead. See Section A.3.4.1 for more information about sizing row caches.

A.3.4 Choosing Memory Location

When you create a row cache or modify a row cache definition, you have the option of specifying where in memory you want Oracle Rdb to create the cache. Row caches can reside in the following memory locations:

- Process global section on OpenVMS and shared memory partition on Digital UNIX.

When you use global sections or shared memory created in the process space, you and other users share virtual memory and the operating system maps a cache to a private address space for each user.

Use the SHARED MEMORY IS PROCESS parameter to specify that the cache be created in a process global section or shared memory partition as shown in the following example:

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD CACHE EMPIDS_LOW_RCACHE
cont> SHARED MEMORY IS PROCESS;
```

This is the default.

- System space buffer

The system space global section is located in the OpenVMS Alpha system space, which means that a system space global section is fully resident, or pinned in memory and does not affect the quotas of the working set of a process.

System space is critical to the overall system. System space buffers are not paged; therefore, they use physical memory, thereby reducing the amount of physical memory available for other system tasks. This may be an issue if your system is constrained by memory. You should be careful when you allocate system space. Nonpaged dynamic pool (NPAGEDYN) and the VMScluster cache (VCC) are some examples of system parameters that use system space.

Use the SHARED MEMORY IS SYSTEM parameter to specify that the cache be created in a system space buffer, as shown in the following example:

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD CACHE EMPIDS_MID_RCACHE
cont> SHARED MEMORY IS SYSTEM;
```

Consider allocating small caches that contain heavily accessed data in system space buffers. When a row cache is stored in a system space buffer, there is no process overhead and data access is very fast because the data does not need to be mapped to user windows. Also, OpenVMS Alpha Version 7 systems and later make additional system space available by moving page tables and balance slots into VLM space. The Hot Row Information screen in the RMU Show Statistics command displays a list of the most frequently accessed rows for a specific row cache.

- Very large memory

Very large memory (VLM) on OpenVMS Alpha systems allows Oracle Rdb to use as much physical memory as is available on your system and to dynamically map it to the virtual address space of database users. VLM provides access to a large amount of physical memory through small virtual address windows. Even though VLM is defined in physical memory, the virtual address windows are defined and maintained in each user's private virtual address space or system space depending on the memory setting.

Use the LARGE MEMORY parameter to specify that the cache be created in large memory.

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD CACHE EMPIDS_OVER_RCACHE
cont> LARGE MEMORY IS ENABLED;
SQL>
```

VLM is useful for large tables with high access rates. The only limiting factor with VLM is the amount of available physical memory on your system.

You view the physical memory through windows. You can specify the number of window panes with the WINDOW COUNT parameter. By default, Oracle Rdb allocates 100 window panes to a process.

Table A-1 summarizes the location in memory of each row cache object and whether process private virtual address windows are needed to access the data.

Table A–1 Memory Locations of Row Cache Objects

SHARED	LARGE	Control Structures	Data Rows	Windows
PROCESS ¹	DISABLED ³	Process global section or shared memory partition	Process global section or shared memory partition	No
PROCESS ¹	ENABLED ⁴	Process global section or shared memory partition	Physical memory	Yes
SYSTEM ²	DISABLED ³	System space	System space	No
SYSTEM ²	ENABLED ⁴	System space	Physical memory	Yes

¹SHARED MEMORY IS PROCESS

- The row cache control structures are located in a process global section or shared memory partition.
- The storage of the data rows depends on whether large memory is enabled or disabled.
 - If large memory is enabled, data is stored in physical memory and windows from each user's process virtual address space are needed to access the data.
 - If large memory is disabled, data is stored in a process global section or shared memory partition and no windows are needed to access the data.

²SHARED MEMORY IS SYSTEM

- The row cache control structures are stored in system space.
- The storage of the data rows depends on whether large memory is enabled or disabled.
 - If large memory is enabled, data is stored in physical memory and windows from each user's process virtual address space are needed to access the data.
 - If large memory is disabled, data is stored in system space and no windows are needed to access the data.

³LARGE MEMORY IS DISABLED

- The storage of the data rows and the row cache control structures depends on whether shared memory is process or system.
 - If shared memory is process, the data and row cache control structures are stored in a process global section or shared memory partition and no windows are needed to access the data.
 - If shared memory is system, the data and row cache control structures are stored in system space and no windows are needed to access the data.

⁴LARGE MEMORY IS ENABLED

- The data rows are stored in physical memory and process private virtual address windows are needed to access the data.
- The storage of the row cache control structures depends on whether shared memory is process or system.
 - If shared memory is process, the control structures are stored in a process global section or shared memory partition.
 - If shared memory is system, the control structures are stored in system space.

It is important to consider the amount of memory available on your system before you start creating and using row caches.

On OpenVMS systems, you can use the DCL command `SHOW MEMORY /PHYSICAL` to check the availability and usage of physical memory. This command displays information on how much memory is used and how much is free. The free memory is available for VLM row caches in addition to user applications.

Because VLM row caches can consume a certain amount of system space for their virtual address windows, Oracle Corporation recommends that you define the VLM row caches first, so that any VLM system space requirements are satisfied before you define system space buffer row caches for small tables that contain frequently accessed data.

The following example shows a system that has 1.5 gigabytes of memory or a total of 196608 OpenVMS Alpha memory pages (an OpenVMS Alpha page is 8192 bytes):

```
$ SHOW MEMORY/PHYSICAL

                System Memory Resources on 29-MAY-1996 21:39:35.40
Physical Memory Usage (pages):      Total      Free      In Use   Modified
Main Memory (1536.00Mb)            196608    183605    12657    346
```

Of the 1.5 gigabytes, 183605 pages remain on the free list. Most of this free memory is available for row cache allocation.

Assume a logical area cache has been defined for the MY_TABLE table. The following SQL statement maps the logical area cache:

```
SQL> ATTACH 'FILE TEST_DB';
SQL> SELECT * FROM MY_TABLE WHERE MY_HASH_INDEX = 100;
```

By issuing this SQL statement, the logical area cache has allocated the necessary memory accounting for 40462 OpenVMS Alpha pages, as shown in the following SHOW MEMORY/PHYSICAL command output:

```
$ SHOW MEMORY/PHYSICAL

                System Memory Resources on 29-MAY-1996 21:46:07.01
Physical Memory Usage (pages):      Total      Free      In Use   Modified
Main Memory (1536.00Mb)            196608    143143    52766    699
```

Notice the amount of free memory has been reduced.

The following SHOW MEMORY/PHYSICAL command was issued after users attached to the database, allocated their working sets, and began to work:

```
                System Memory Resources on 29-MAY-1996 23:48:06.67
Physical Memory Usage (pages):      Total      Free      In Use   Modified
Main Memory (1536.00Mb)            196608    81046     112498   3064
```

In this example, only 81046 OpenVMS Alpha pages are left on the free list.

A.3.4.1 Sizing Considerations

The following information is intended to help you determine in which memory location to place your cache based on system resources. Generally, if your cache will fit into a process global section or system space buffer, then it will perform slightly better. If space is an issue, then you should place the cache in VLM.

When a cache is created in a process global section or system space buffer, Oracle Rdb sizes it using the following values:

- Each slot requires 48 bytes plus the length of the slot rounded to the next 4-byte boundary.
- Each cache requires a hash table of (4 * (the number of cache slots rounded to the next higher power of 2)) bytes.
- Each cache requires (24 * the maximum number of users) bytes.

When a cache is created in VLM, Oracle Rdb sizes it using the following values:

- Each slot requires 24 bytes plus the length of the slot rounded up to the next 4-byte boundary.

When VLM is enabled and the cache is created in a process global section or system buffer space, Oracle Rdb sizes it using the following values:

- Each slot requires 24 bytes.
- Each cache requires a hash table of $(4 * (\text{the number of cache slots rounded up to the next higher power of 2}))$ bytes.
- Each cache requires $(24 * \text{the maximum number of users})$ bytes.

The following example shows how Oracle Rdb sizes a cache containing 150,000 slots with a slot size of 500 bytes in a process global section or system space buffer and a maximum of 350 users. (Note that 2 to the 17th power is 262144.)

Example A-1 Sizing a Row Cache in a Global Section or System Space Buffer

```
Total
number = (150000*(500+48)) + (262144*4) + (24*350)
of
bytes
= 83,256,976 bytes
```

The following example shows how Oracle Rdb sizes the same cache in VLM.

Example A-2 Sizing a Row Cache in VLM

```
Total
number = (150000*(500+24))
of
bytes
= 78,600,000 bytes
```

The following example shows how Oracle Rdb sizes the same cache in a process global section or system space buffer with VLM enabled.

Example A-3 Sizing a Row Cache in Memory with VLM Enabled

```
Total
number = (150000*24) + (262144*4) + (24*350)
of
bytes
= 4,656,976 bytes
```

A.4 Using Row Cache

The following sections describe how to set parameters for the row cache feature.

A.4.1 Enabling and Disabling Row Cache

There are three ways in which Row Caching can be enabled and/or disabled.

1. You can enable row caching for a database by using the ROW CACHE IS ENABLED clause of the SQL ALTER DATABASE and CREATE DATABASE statements. The following example shows how to enable the row cache feature and its requirements:

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> NUMBER OF CLUSTER NODES IS 1
cont> JOURNAL ENABLED (FAST COMMIT ENABLED)
cont> RESERVE 20 CACHE SLOTS
cont> ROW CACHE IS ENABLED;
```

You can disable row caching for a database by using the ROW CACHE IS DISABLED clause of the SQL ALTER DATABASE and CREATE DATABASE statements:

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ROW CACHE IS DISABLED;
```

Row caching is also disabled if one of the conditions described in Section A.2 becomes false.

When row caching is disabled, all previously created and assigned row caches remain in existence for future use when row caching is enabled again.

The database must be closed when you enable or disable row caching.

2. The RMU/SET command allows you to enable or disable row caching using an unjournalled operation. This is needed to disable row caches if you have system tables mapped to row caches and you need to perform SQL operations that require exclusive database access.

```
RMU/SET/ROW_CACHE[/DISABLED|/ENABLED] database_name
```

For example, adding a row cache to a database requires exclusive database access. Execute this command before adding a new row cache using SQL then re-enable row caching.

3. The RMU/OPEN/ROW_CACHE=DISABLED command is used to keep row cache enabled in the database but not used for the duration of the open. This is necessary in order to set up row caching in a Hot Standby environment. Row caching is not allowed to be active on the standby database. Therefore, this command should be issued on the standby system to open the database without activating row caching.

A.4.2 Specifying Checkpointing and Sweeping Options

The following sections provide guidelines for specifying checkpointing and sweeping options.

A.4.2.1 Choosing the Checkpoint Source and Target Options

For greatest flexibility, provide each row cache with its own checkpoint source and target options as follows:

- The source rows to read
This determines which source rows in the cache to write back to disk. Only updated rows or all rows can be selected. By default, only updated rows are selected.
- The target location to write the rows

This determines whether the source rows are written back to the database pages or written out to a separate row cache backing file.

You can specify the target location using the following parameters of the ADD, ALTER, and CREATE CACHE clauses. Note that you cannot specify that all rows are checkpointed to the database.

- CHECKPOINT UPDATED ROWS TO BACKING FILE
- CHECKPOINT UPDATED ROWS TO DATABASE
- CHECKPOINT ALL ROWS TO BACKING FILE

The following table lists the advantages and disadvantages of each checkpoint target:

Table A-2 Checkpoint Target Options

Advantages	Disadvantages
Checkpoint to Database	
Does not require any more disk space.	Is slower due to contention for database page buffers.
Simpler to understand because it uses the traditional database page buffers.	Upon node failure, the row cache is not re-populated.
Unmarks slots in the row cache so they can be reused for other rows.	Greater conflict with other users since row and page locks are maintained. The row cache server (RCS) process does not respond to requests to release row or page locks
Writing back to database pages reclaims space on database pages from erased or modified rows that have been reduced in size.	
Checkpoint to Backing File	
Can checkpoint all rows allowing a way to repopulate row caches that are predominantly read-only while recovering from a node failure.	Requires extra disk space to create two backing files per cache.
Faster at writing sequential I/O operations to backing file.	Only used for node failure protection.
Can be placed on different spindles so that other database I/O activity will not be impacted.	Marked rows tend to stay marked. By definition, rows in a row cache are only unmarked when they are written back to the database.
Used upon node failure to repopulate the row cache.	Space on the database pages resulting from erased rows and modified rows that are reduced in size is not reclaimed.

A.4.2.2 Choosing the Checkpoint Interval

You must specify a checkpoint interval in the following way: use the CHECKPOINT TIMED EVERY *s* SECONDS parameter of the ROW CACHE IS ENABLED clause. This checkpoint parameter applies to the RCS process only.

This value can be overridden by the RDM\$BIND_CKPT_TIME logical (this logical is also used to override the FAST COMMIT checkpoint interval). If nothing is specified, Oracle Rdb uses a default checkpoint interval of 15 minutes.

A.4.2.3 Specifying Sweeping Parameters

You set the number of updated rows that will be swept by using the NUMBER OF SWEEP ROWS IS parameter of the ADD, ALTER, and CREATE CACHE clause.

```
SQL> ALTER DATABASE FILENAME INVENTORY
cont> ALTER CACHE PARTS
cont> ROW LENGTH IS 104 BYTES
cont> CACHE SIZE IS 2000 ROWS
cont> CHECKPOINT ALL ROWS TO BACKING FILE
cont> NUMBER OF SWEEP ROWS IS 200;
```

A row in a row cache cannot be reused if it is marked (modified) or if its reference count is greater than zero. In the latter case, one or more users have a reference to this row in their row cache working sets. The RCS sweep operation tries to eliminate these restrictions from rows in the row cache so these rows can be reused to insert new rows.

The RCS process writes committed modified rows back to the database, up to a maximum of the NUMBER OF SWEEP ROWS defined for the row cache. It is important that this value be set properly so that when a sweep is initiated, the RCS process clears out enough slots to allow sufficient insertion activity before another sweep operation is necessary. Typically, a value of 10 percent to 30 percent of the size of the row cache would be sufficient. Make sure that the sweep count is larger than the value of the row cache's reserved count, specified by the NUMBER OF RESERVED ROWS IS *N* clause.

You can override the row cache's defined sweep count value by defining the RDM\$BIND_RCS_SWEEP_COUNT logical name. Note, however, the value of this logical name applies to all row caches.

During a sweep operation, the RCS process may also initiate a dialogue with current users to reset the reference counts of the rows in the cache. The RCS process will only do this during a sweep operation if the number of database recovery processes since the last sweep operation of this row cache has exceeded the number specified by the RDM\$BIND_RCS_CLEAR_GRICS_DBR_CNT logical name. Only processes that have abnormally terminated fail to clean up their reference counts normally.

An RCS sweep operation is triggered when a row cache is considered "clogged". A row cache is considered clogged when a user fails to find any available slots in which to insert rows. Even after a row cache is considered full, a user may still be able to insert rows into that row cache if the user still has reserved slots to use.

The RCS process clears the clogged flag if the sweep operation was successful in opening up some slots. The clogged flag can also become clear during a checkpoint operation if the RCS process has detected row cache entries with zero reference counts. This will only happen if the clogged flag stays set for three consecutive checkpoint operations.

A.4.2.4 Specifying the Size and Location of the Cache Backing File

When allocating the size of the cache backing (.RDC) files, consider the following:

- Whether all rows or only marked rows will be checkpointed
- The amount of update activity in the row cache
- Whether you want to create new backing files on each database open or re-use existing backing files

If you want Oracle Rdb to automatically rebuild an entire row cache in memory after a node failure, then define the row cache to checkpoint all rows to a cache backing file. If you want Oracle Rdb to repopulate the row cache with only the rows that were modified at the time, then define the row cache to checkpoint only updated rows to the cache backing file.

The decision you make determines how to size the cache backing files.

If all rows are to be checkpointed, use the following formula to determine the number of blocks to allocate for the cache backing file.

$$\begin{array}{l} \text{Number of} \\ \text{blocks} \end{array} = (\text{slot count} * (\text{row length} + 40)) / 512 \text{ bytes per block}$$

If only the updated rows are to be written to the backing file, use the following formula to allocate the backing file, based on the estimated number of updated rows in the row cache.

$$\begin{array}{l} \text{Number of} \\ \text{blocks} \end{array} = (\# \text{ of updated rows} * (\text{row length} + 40)) / 512 \text{ bytes per block}$$

You can overwrite the allocation specified in the row cache definition with the RDM\$BIND_CKPT_FILE_SIZE system logical name. This specifies the percentage of the row cache size to allocate for the backing file. The default is 40 percent.

$$\begin{array}{l} \text{Number of} \\ \text{blocks} \end{array} = (0.40 * \text{slot count} * (\text{row length} + 40)) / 512 \text{ bytes per block}$$

When checkpointing to backing files, Oracle Rdb needs two backing files for each cache. One is used for the last checkpoint (committed rows), and the other is for the current checkpoint. Make sure there is enough disk space for two backing files for each cache. By default, Oracle Rdb deletes the backing files upon successful database shutdown and recreates them when the database is reopened. If you prefer, you can tell Oracle Rdb to save the backing files and re-use them on the subsequent database open by defining the system logical RDM\$BIND_RCS_KEEP_BACKING_FILES to "1".

If you are checkpointing a row cache to the database, you do not need to specify an allocation or location for the cache backing file. Oracle Rdb will ignore these clauses.

If you have a read-only cache, specify 1 block for the size of the cache backing file as follows:

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD CACHE RCACHE_2
cont> LOCATION IS WORK$DISK1:[RCS]
cont> ALLOCATION IS 1 BLOCK;
```

A.4.3 Controlling What is Cached in Memory

The ROW REPLACEMENT parameter of the ADD, ALTER, and CREATE CACHE clause gives you some control over what happens when a row cache becomes full. If row replacement is enabled for a particular row cache, new rows will replace the oldest, unused, unmarked rows once the cache is full. If row replacement is disabled, new rows are not placed in the cache once the cache is full; they will always be retrieved from disk.

When you use the ROW REPLACEMENT IS DISABLED parameter, the data that was memory resident stays that way and therefore all subsequent reads occur from memory rather than disk.

You can increase performance by making the following types of rows memory resident.

- Nonleaf nodes of a B-tree index
Be sure to account for the nodes splitting when you specify the size for the row cache. If a parent node splits and there is no room in the cache for the new node, the new node will not be held in memory.
- Data that is primarily read-only
Data that does not change very often, such as dimension tables in a data warehouse environment, is a good candidate for keeping resident in memory.
- Data that is update-intensive; when the entire table can fit in the cache
Oracle Rdb optimizes access when the cache is defined with row replacement disabled.

Enabling row replacement is beneficial when access patterns of a table are random. This ensures that the most frequently accessed rows remain in memory. Often, there may not be enough physical memory to cache an entire table, so caching the most frequently used rows can improve performance.

A.4.3.1 Row Replacement Strategy

Global and local buffers use the least-recently used (LRU) replacement strategy for database pages. Row caching uses a modified form of the LRU replacement strategy. Each database user can protect the last 10 rows they accessed. This group of rows is referred to as a **working set**. Rows that belong to a working set are considered to be **referenced** and are not eligible for row replacement. Any row that is in a cache and is not part of a working set is considered an **unreferenced** row. The unreferenced rows are eligible for replacement if they are not marked.

A.4.3.2 Inserting Rows into a Cache

Each user process requests rows from the database. A user process, which reads a row from a storage area, tries to insert the row into the cache (if it is not already there). If a slot is available, the requested row is stored in the cache, if it fits. If no more slots are available in the cache, one of the following happens:

- If ROW REPLACEMENT IS ENABLED, and an unmarked, unreferenced row can be found, that row is replaced by the new row. Oracle Rdb chooses the unreferenced row randomly.
- If ROW REPLACEMENT IS DISABLED, the row is not stored in the cache. This means that when the cache fills, it will not accept new rows. Reserved slots, however, can still be used.

You can prevent individual processes from inserting new rows into any Oracle Rdb row cache by defining the process logical `RDMS$BIND_RCACHE_INSERT_ENABLED` to "0". When defined, a process can only use what already exists in the row caches; the process cannot insert a row into a row cache. This option is useful if, for example, you want to keep nightly batch processes that perform large reporting functions from filling up row caches that are also used by the more important, daily, on-line transaction processing servers.

If system usage is lighter at night, you may want to preload row caches so that the data is available in memory during the day when database activity is at its peak.

The remainder of this section illustrates how Oracle Rdb inserts rows into a cache.

The example makes the following assumptions:

- Row caching is enabled.
- Row replacement is enabled.
- A row cache (`RCACHE_1`) has been created with 25 slots.
- Two processes (Jones and Smith) are attached to the database.
- The rows in the row cache are not modified.

The initial allocation is as follows:

Row Cache RCACHE_1

Slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	
Row																										
Counter																										

Working Set of Process Jones

Slot	1	2	3	4	5	6	7	8	9	10
Row										

Working Set of Process Smith

Slot	1	2	3	4	5	6	7	8	9	10
Row										

NU-3614A-RA

1. Process Jones executes a query that causes 5 rows to be read into the first 5 slots of the row cache.

Slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Row	A	B	C	D	E																				
Counter	1	1	1	1	1																				

Working Set of Process Jones

Slot	1	2	3	4	5	6	7	8	9	10
Row	A	B	C	D	E					

NU-3615A-RA

Each row slot has a working set counter associated with it. The working set counter indicates whether the row belongs to a working set. A positive value indicates that the row belongs to a working set. If a row belongs to a working set, it is not eligible for row replacement.

- Process Smith requests 15 rows from the database. The first 10 rows requested go into Smith's working set as follows:

Working Set of Process Smith

Slot	1	2	3	4	5	6	7	8	9	10
Row	F	G	H	I	J	K	L	M	N	O

NU-3616A-RA

Process Smith's working set has exactly 10 slots, and all 10 are being used. The least recently used row is replaced by the eleventh row that Process Smith reads into the cache. Rows 12 through 15 also overwrite the contents of slots 2 through 5 respectively.

After the 15 rows are read into the cache, the cache appears as follows:

Slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Row	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T					
Counter	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1					

NU-3617A-RA

After the 15 rows are read into the cache, Process Smith's working set appears as follows:

Working Set of Process Smith

Slot	1	2	3	4	5	6	7	8	9	10
Row	P	Q	R	S	T	K	L	M	N	O

NU-3618A-RA

At this point, rows F, G, H, I, and J are unreferenced. They are in the cache but they do not belong to the working set of any process. Oracle Rdb sets the working set counter for an unreferenced row to zero. The unreferenced rows are eligible for replacement if they have not been modified and row replacement is enabled. Any process can read rows F, G, H, I, or J without executing an I/O operation. However, if a process requires a row that is not currently in the cache, one of the rows F, G, H, I, or J is replaced with the new row.

Each slot in the row cache contains a modification flag. If the row has been modified, but not yet flushed to disk, it is considered to be **dirty**. Dirty rows are not candidates for row replacement either. Modified rows are written to disk by the row cache server (RCS) process. See Section A.4.2.1 for more information.

- Process Jones requests 7 more rows: M, U, V, W, X, Y, and Z. Jones can read row M without performing any I/O because M is already in the cache. An additional slot does not get filled in the row cache, but row M is added to Process Jones' working set.

Process Jones' working set now appears as follows:

Working Set of Process Jones

Slot	1	2	3	4	5	6	7	8	9	10
Row	Y	B	C	D	E	M	U	V	W	X

NU-3619A-RA

Rows U, V, W, X, and Y go into the remaining slots in the row cache and the row cache appears as follows:

Slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Row	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
Counter	0	1	1	1	1	0	0	0	0	0	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1

NU-3620A-RA

Note that the working set counter for slot 13 indicates that row M is in two working sets. This indicates that two processes are accessing the same row. The number of processes sharing a particular slot is known as the **share count**.

At this point, the cache is full. If row replacement were disabled for the row cache, then row Z could not be inserted. However, in this example, row replacement is enabled, and there is an unreferenced slot. Therefore, Oracle Rdb will choose an unreferenced slot to make room for the new row, Z. (In this example, the unreferenced slots are A, F, G, H, I, and J.)

A.5 Examining Row Cache Information

You can display the attributes using the SHOW CACHE statement as in the following example:

```
SQL> SHOW CACHE PARTS;
PARTS
      Cache Size:      204 rows
      Row Length:      104 bytes
      Row Replacement: Enabled
      Shared Memory:   Process
      Large Memory:    Disabled
      Window Count:    100
      Reserved Rows:   20
      Sweep Rows:      1004
      Allocation:      100 blocks
      Extent:          100 blocks
```

You can also use the RMU Dump command with the Header qualifier to display row cache information, as in the following example:

Example A-4 Row Cache Parameters

```
$ RMU/DUMP/HEADER INVENTORY
.
.
.
Row Caches...      1
- Active row cache count is 4
- Reserved row cache count is 20
- Checkpoint information
  Time interval is 10 seconds
  Default source is updated rows
  Default target is backing file
  Default backing file directory is "DISK1:[RDB]"
.
.
.
Row cache "PARTS"
Cache ID number is 4 2
Allocation...     3
- Row slot count is 204
- Maximum row size allowed in cache is 104 bytes
- Working set count is 10
- Maximum slot reservation count is 20
- Row replacement is enabled
Sweeping...      4
- Sweep row count is 1004
- Maximum batch I/O count is 0
Checkpointing... 5
- Source is updated rows (database default)
- Target is backing file (database default)
- No checkpoint information available
- Checkpoint sequence is 0
Files...         6
- Default cache file directory is "DISK1:[RDB]"
- File allocation is 100 blocks
- File extension is 100 blocks
Hashing...       7
- Hash value for logical area DBIDs is 211
- Hash value for page numbers is 11
Shared Memory... 8
- System space memory is disabled
- Large memory is disabled
- Large memory window count is 100
Cache-size in different sections of memory... 9
- Without VLM, process or system memory requirement
  is 309760 bytes
- With VLM enabled...
  - Process or system memory requirement is 38768 bytes
  - Physical memory requirement is 280000 bytes
  - VLM Virtual memory address space requirement is
  approximately 102400 bytes
.
.
.
```

The following callouts identify the parameters in Example A-4:

- 1 Row Caches . . .
 - Active row cache count is 4

This specifies the number of row caches currently defined in this database.

- Reserved row cache count is 20

This specifies the number of slots that are available in the database. The cache slots are reserved with the `RESERVE n CACHE SLOTS` parameter of the `ALTER` or `CREATE DATABASE` statements.

- Checkpoint information

This displays database-level checkpoint information specified using parameters of the `ADD`, `ALTER`, or `CREATE CACHE` clauses.

- Time interval is 10 seconds

A checkpoint is one full pass through all active row caches, attempting to write all or just marked rows back to their respective storage areas or the backing file. The time interval is set with the `CHECKPOINT TIMED EVERY s SECONDS` parameter.

- Default source is updated rows

Only updated rows are written to the backing file or back to the database storage areas.

- Default target is backing file

Specifies that the default target for the checkpoint is the backing file and not the database. This is the default target when the `CHECKPOINT UPDATED ROWS` parameter is not set.

- Default backing file directory is “`DISK1:[RDB]`”.

The default cache file directory is the directory where Oracle Rdb places the cache backing store files. If you do not explicitly include a directory specification, Oracle Rdb will place the backing file in the directory where the database root file is stored.

2 Cache ID number is

Oracle Rdb assigns an ID to each defined row cache in the database.

3 Allocation . . .

- Row slot count is 204

This is specified with the `CACHE SIZE IS n ROWS` parameter.

- Maximum row size allowed in cache is 104 bytes

This is specified with the `ROW LENGTH IS n BYTES` parameter.

- Working set count is 10

This is the number of “in use” rows that are not eligible for row replacement.

- Maximum slot reservation count is 20

This is specified with the `NUMBER OF RESERVED ROWS` parameter. The default value is 20 rows.

The number of reserved rows indicates how many slots in the cache Oracle Rdb will reserve for each process. Reserving many rows minimizes row cache locking while rows are inserted into the cache.

The number of reserved rows parameter is also used when searching for available slots in a row cache. The entire row cache is not searched on the initial pass. This parameter is used as the maximum number of rows that are searched for a free slot. If at least one free slot is found, the insert operation can proceed. If no free slots are found in this initial search, Oracle Rdb will continue searching through the cache until it finds a free slot.

- Row replacement is enabled
This is specified with the ROW REPLACEMENT parameter. Row replacement is enabled by default.

4 Sweeping . . .

- Sweep row count is
Sets the number of marked rows that will be swept back to the database or backing file when the row cache is full and a user attempts to find an empty slot.

5 Checkpointing . . .

- Source is updated rows (database default)
The source of updated rows is the same as the database default.
- Target is backing file (database default)
The target for marked rows is the database default.

6 Files . . .

- Default cache file directory is "DISK1:[RDB]"
The LOCATION parameter specifies a directory specification for the cache backing store file. Oracle Rdb writes to the cache backing store file when the RCS process checkpoints. Oracle Rdb automatically generates a file name with a file extension of .rdc. The default location for the cache backing store file is the directory where the database root file is stored.
The LOCATION parameter can be specified at the database level or at the row cache level. If you include the LOCATION parameter in the ADD CACHE or CREATE CACHE clauses of the CREATE or ALTER DATABASE statements, the directory you specify becomes the default directory location for all the row caches that are defined for the database. You can, however, override the default directory location for individual row caches by specifying the LOCATION parameter in the row cache definition.
- File allocation is 100 blocks
The ALLOCATION parameter specifies the initial size of the cache backing file. The default allocation is 40 percent of the cache size. The cache size is determined by multiplying the number of rows in the cache by the row length.
- File extension is 100 blocks
The EXTENT parameter specifies the number of pages by which the cache backing store file can be extended after the initial allocation has been reached. The default extent is 127 multiplied by the number of rows in the cache.

7 Hashing . . .

- Hash value for logical area DBIDs is 211
- Hash value for page numbers is 11

The hash values are used by Oracle Rdb to fine-tune the distribution of hash table queues in the row cache.

8 Shared Memory . . .

- System space memory is disabled

This is specified with the SHARED MEMORY parameter. This specifies whether Oracle Rdb creates the row cache in shared memory. The row cache is created in a process global section (OpenVMS) or in a shared memory partition (Digital UNIX) by default.

- Large memory is disabled

This is specified with the LARGE MEMORY parameter. This specifies whether Oracle Rdb creates the row cache in physical memory. Large memory is disabled by default.

- Large memory window count is 100

This is specified with the WINDOW COUNT parameter. The default value is 100 windows. The WINDOW COUNT specifies how many locations of the physical memory are mapped to each user's private window in virtual address space.

9 Cache-size in different sections of memory . . .

- Without VLM, process or system memory requirement is 309760 bytes
When the cache is created in a process global section or system space buffer and VLM is not enabled, this is the memory requirement.

- With VLM enabled . . .

- Process or system memory requirement is 38768 bytes

When VLM is enabled and the cache is created in a process global section or system space buffer, this is the memory requirement.

- Physical memory requirement is 280000 bytes

The actual cached data requires this space in VLM.

- VLM Virtual memory address space is approximately 102400 bytes

This is the address space used by the virtual memory windows.

A.5.1 RMU Show Statistics Screens and Row Caching

The RMU Show Statistics command displays much information regarding row caches. The following are titles of some of the screens that can be displayed regarding row cache:

- Summary Cache Statistics
- Summary Cache Unmark Statistics
- Row Cache (One Cache)
- Row Cache (One Field)
- Row Cache Utilization

- Hot Row Information
- Row Cache Status
- Row Cache Queue Length
- Row Length Distribution
- RCS Statistics
- Row Cache Dashboard
- RCS Dashboard
- Per-Process Row Cache Dashboard

A.6 Examples

This section includes some practical examples on using the row cache feature of Oracle Rdb.

A.6.1 Loading a Logical Area Cache

Use the following steps to place an entire table in a row cache:

1. Determine how many rows are in the table.

```
SQL> SELECT COUNT(*) FROM EMPLOYEES;
          100
1 row selected
```

2. Create a logical cache large enough to hold to the table.

Use the table name as the name of the cache to create the logical cache. Oracle Rdb will determine the row length from the table.

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD CACHE EMPLOYEES
cont> CACHE SIZE IS 100 ROWS;
```

3. Cause Rdb to sort the table by an indexed field.

This causes rows to be read by DBKEY after the sort is complete.

```
SQL> SELECT * FROM EMPLOYEES ORDER BY EMPLOYEE_ID;
EMPLOYEE_ID  LAST_NAME      FIRST_NAME  MIDDLE_INITIAL
ADDRESS_DATA_1  ADDRESS_DATA_2  CITY
STATE  POSTAL_CODE  SEX  BIRTHDAY  STATUS_CODE
00197      Danzig      Chris      NULL      Acworth
136 Beaver Brook Circle
NH      03601      F      21-Jun-1939  1
.
.
.
```

A.6.2 Caching Database Metadata

Because metadata is frequently accessed, you may want to cache some or all of your database's metadata. You can map the entire contents of the RDB\$SYSTEM storage area to a physical area row cache. Alternatively, you can map certain system tables, such as RDB\$RELATIONS and RDB\$INDICES, into separate logical area row caches.

To do this, follow these steps.

1. Use the RMU/DUMP/AREA command to display the contents of the storage area. (Note that the RMU Dump command output uses the term records to refer to rows.)

```
$RMU/DUMP/AREA=RDB$SYSTEM/OUT=RMU_DUMP_1.OUT MF_PERSONNEL
$SEARCH/STATISTICS RMU_DUMP_1.OUT "RECORD LENGTH", "STATIC_DATA"

          00A2 0050 record length 162 bytes
          00E8 008B record length 232 bytes
          00C4 00C6 record length 196 bytes
          00E4 0101 record length 228 bytes
          0088 013C record length 136 bytes
          023C 0177 record length 572 bytes
          0220 01B2 record length 544 bytes
          030C 01ED record length 780 bytes

          .
          .
          .
Files searched:          1          Buffered I/O count:          100
Records searched:        62260       Direct I/O count:          441
Characters searched:    3459752      Page faults:              20
Records matched:        96           Elapsed CPU time: 0 00:00:01.63
Lines printed:          96           Elapsed time: 0 00:00:02.83
```

2. Determine the row length and slot count.

Keep in mind that other structures may be stored in this area because it can be specified as the default storage area for Oracle Rdb.

3. Add the physical cache and assign it to the RDB\$SYSTEM storage area.

In the following example, row length has been rounded up and the cache size has been increased to allow for future growth.

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD CACHE RDB_SYSTEM_CACHE
cont> CACHE SIZE IS 9000 ROWS
cont> ROW LENGTH IS 800 BYTES;
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ALTER STORAGE AREA RDB$SYSTEM
cont> CACHE USING RDB_SYSTEM_CACHE;
```

4. Or, add the logical area caches to the Rdb system tables of interest.

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> ADD CACHE RDB$RELATIONS
cont> CACHE SIZE IS 1000 ROWS
cont> ROW LENGTH IS 500 BYTES
cont> ADD CACHE RDB$INDICES
cont> CACHE SIZE IS 2000 ROWS
cont> ROW LENGTH IS 500 BYTES;
```

When caching metadata, you will experience conflicts when executing database operations through SQL that require exclusive database access. For example, adding new row caches or dropping existing ones requires exclusive database access. When the SQL command is parsed, the Oracle Rdb system tables are queried. This access to the system tables creates the row caches and causes the RCS process to come up to manage those row caches. As a result, the database now has another “user”, the RCS process. This causes the exclusive database operation to fail.

To resolve this, you must first turn off row caching temporarily using the RMU Set command specifying the Row_Cache and Disabled qualifiers. Then, perform the SQL operation that requires exclusive database access. Finally, re-enable row caching using the RMU Set command with the Row_Cache and Enabled qualifiers.

A.6.3 Caching a Sorted Index

To cache a sorted index, use the following steps:

1. Display the number of index nodes using the RMU Analyze Index command. (Note that the RMU Analyze command uses the term records to refer to rows.)

```
$RMU/ANALYZE/INDEX MF_PERSONNEL EMP_LAST_NAME  
  
Index EMP_LAST_NAME for relation EMPLOYEES duplicates allowed  
Max Level: 2, Nodes: 8, Used/Avail: 1625/3184 (51%), Keys: 90, Records: 67  
Duplicate nodes: 16, Used/Avail: 264/312 (85%), Keys: 16, Records: 33
```

2. Count the number of nodes and duplicate nodes.
3. Allocate slots based on the number of nodes currently used and allow for future growth.

In this example, allocating 28 slots would be reasonable.

4. Determine node and duplicate node size. Sorted indexes with duplicates should be sized at 430 bytes rounded up to the next 4-byte interval.
5. Create a logical cache for the sorted index.

```
SQL> ALTER DATABASE FILENAME MF_PERSONNEL  
cont> ADD CACHE EMP_LAST_NAME  
cont> ROW LENGTH IS 440 BYTES  
cont> CACHE SIZE IS 28 ROWS;
```

Row Cache Statements

B.1 ALTER DATABASE Statement

B.1.1 Overview

Alters a database in any of the following ways:

- For single-file and multifile databases, the ALTER DATABASE statement changes the characteristics of the database root file.

The ALTER DATABASE statement lets you override certain characteristics specified in the database root file parameters of the CREATE DATABASE statement, such as whether or not a snapshot file is disabled. In addition, ALTER DATABASE lets you control other characteristics you cannot specify in the CREATE DATABASE database root file parameters, such as whether or not after-image journaling is enabled.

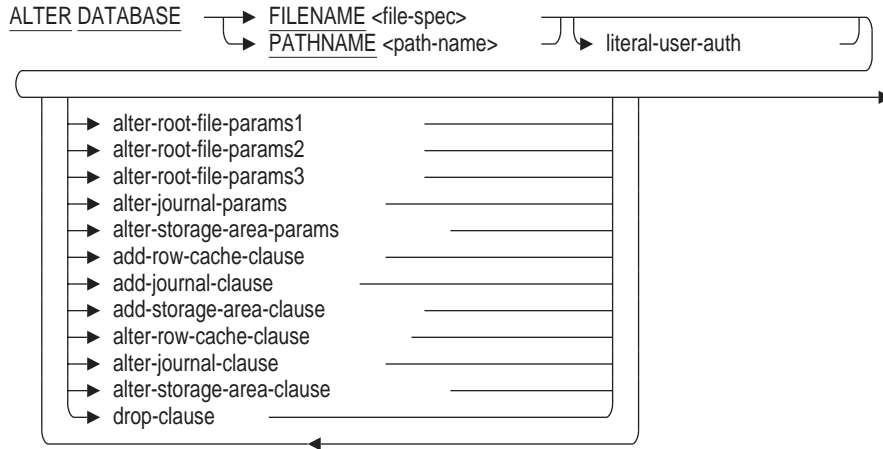
- For single-file and multifile databases, the ALTER DATABASE statement changes the storage area parameters.
- For multifile databases *only*, the ALTER DATABASE statement adds, alters, or deletes storage areas.

B.1.2 Environment

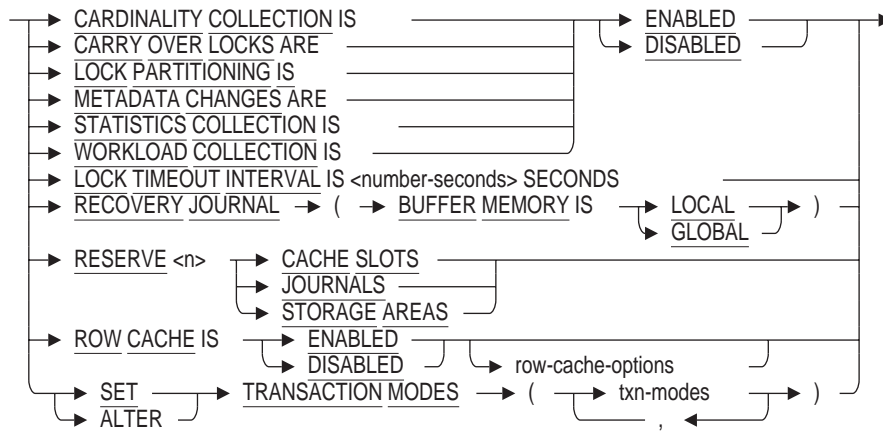
You can use the ALTER DATABASE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

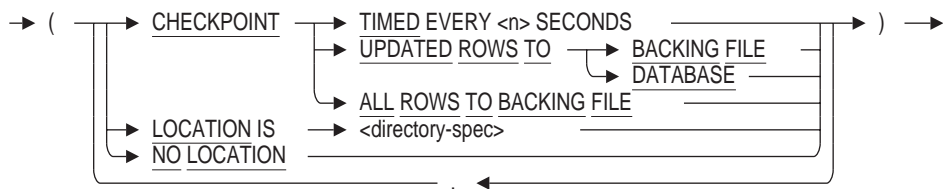
B.1.3 Format



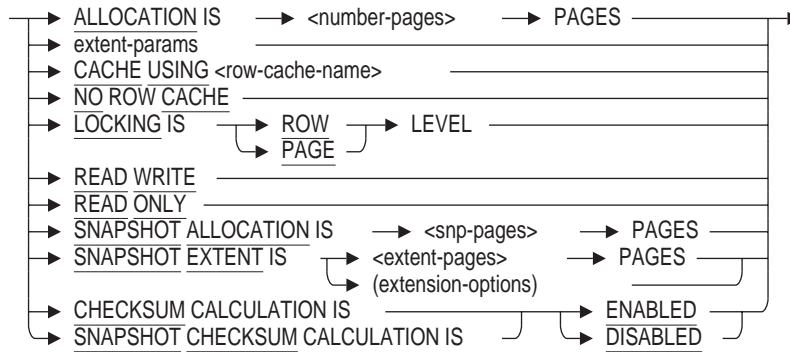
alter-root-file-params2 =



row-cache-options =



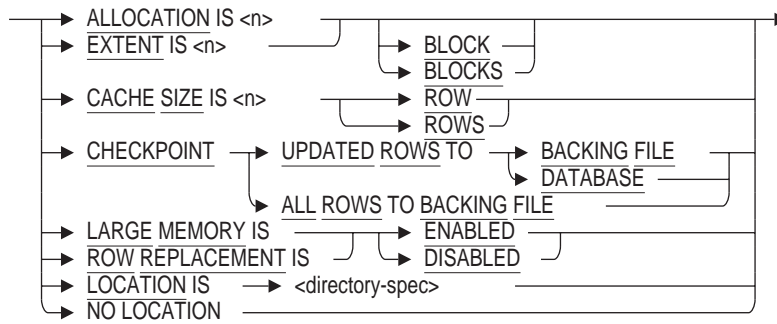
alter-storage-area-params =



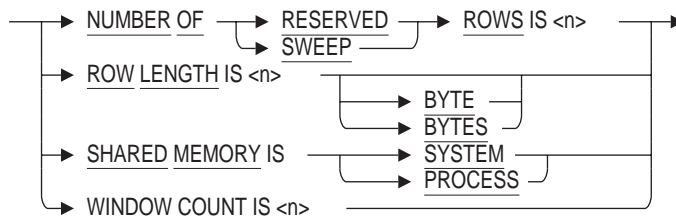
add-row-cache-clause =



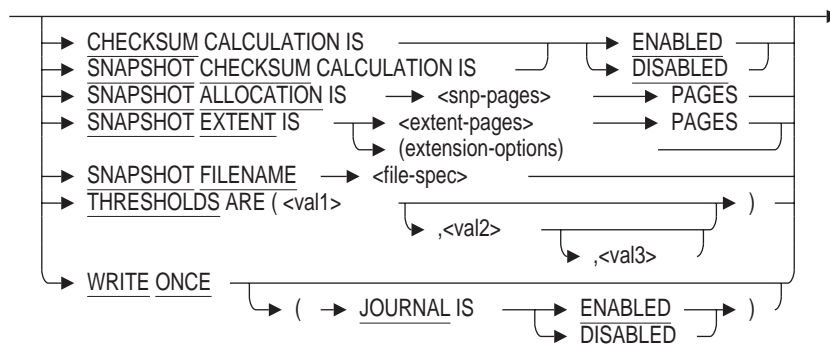
row-cache-params1 =



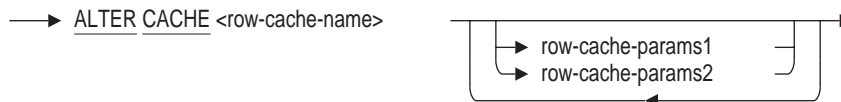
row-cache-params2 =



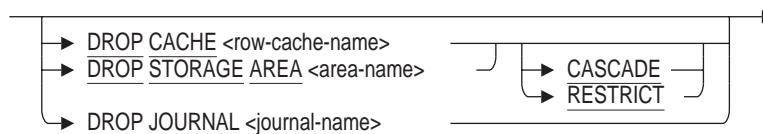
storage-area-params-2 =



alter-row-cache-clause =



drop-clause =



B.1.4 Arguments

B.1.4.1 RECOVERY JOURNAL (BUFFER MEMORY IS {LOCAL | GLOBAL})

The RUJ buffers used by each process are normally allocated in local virtual memory. With the introduction of ROW CACHE, these buffers can now be assigned to a shared global section (GLOBAL memory) so that the recovery process can process this in memory buffer and possibly avoid a disk access.

This buffer memory can be defined a GLOBAL to improve ROW CACHE performance for recovery. If ROW CACHE is DISABLED then buffer memory is always LOCAL.

B.1.4.2 RESERVE n CACHE SLOTS

Specifies the number of row caches for which slots are reserved in the database.

You can use the RESERVE CACHE SLOTS clause to reserve slots in the database root file for future use by the ADD CACHE clause. Row caches can be added only if there are row cache slots available. Slots become available after a DROP CACHE clause or a RESERVE CACHE SLOTS clause.

The number of reserved slots for row cache cannot be decreased once the RESERVE clause is issued. If you reserve 10 slots and later reserve 5 slots, you have a total of 15 reserved slots for row caches.

Reserving row cache slots is an offline operation (requiring exclusive database access).

B.1.4.3 CACHE USING row-cache-name

Assigns the named row cache as the default physical row cache for all storage areas in the database. All rows stored in each storage area, whether they consist of table data, segmented string data, or special rows such as index nodes, are cached.

The row cache must exist before terminating the ALTER DATABASE statement.

Alter the database and storage area to assign a new physical area row cache to override the database default physical area row cache. Only one physical area row cache is allowed for each storage area.

You can have multiple row caches containing rows for a single storage area by defining logical area row caches, where the row cache name matches the name of a table or index.

If you do not specify the CACHE USING clause or the NO ROW CACHE clause, NO ROW CACHE is the default for the database.

B.1.4.4 NO ROW CACHE

Specifies that the database default is not to assign a row cache to all storage areas in the database. You cannot specify the NO ROW CACHE clause if you specify the CACHE USING clause.

Alter the storage area and name a row cache to override the database default. Only one row cache is allowed for each storage area.

If you do not specify the CACHE USING clause or the NO ROW CACHE clause, NO ROW CACHE is the default for the database.

B.1.4.5 ROW CACHE IS ENABLED/ROW CACHE IS DISABLED

Specifies whether or not you want Oracle Rdb to enable the row caching feature.

Enabling cache support does not affect database operations until a cache is created and assigned to one or more storage areas.

When the row caching feature is disabled, all previously created and assigned caches remain in existence for future use when the row caching feature is enabled.

Enabling and disabling the row cache feature is an offline operation (requiring exclusive database access).

B.1.4.5.1 CHECKPOINT TIMED EVERY N SECONDS Specifies the frequency with which the RCS process checkpoints the contents of the row caches back to disk. **The RCS process does not use the checkpoint frequency options of the FAST COMMIT clause.**

The frequency of RCS checkpointing is important in determining how much of an AIJ file must be read during a REDO operation following a node failure. It also affects the frequency that marked records get flushed back to the database, for those row caches that checkpoint to the database. The default is every 15 minutes (900 seconds).

B.1.4.5.2 CHECKPOINT ALL ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO DATABASE Specifies the default source and target for checkpoint operations for all row caches. If ALL ROWS is specified, then the source records written during each checkpoint operation are both the modified and the unmodified rows in a row cache. If UPDATED ROWS is specified, then just the modified rows in a row cache are checkpointed each time.

If the target of the checkpoint operation is BACKING FILE, then the RCS process writes the source row cache entries to the backing (.rdc) files. The row cache LOCATION, ALLOCATION, and EXTENT clauses are used to create the backing files. Upon recovery from a node failure, the database recovery process is able to re-populate the in-memory row caches from the rows found in the backing files.

If the target is DATABASE, then the target rows (only UPDATED ROWS is allowed) are written back to the database. The row cache LOCATION, ALLOCATION, and EXTENT clauses are ignored. Upon recovery from a node failure, the database recovery process has no data on the contents of the row cache. Therefore, it does not re-populate the in-memory row caches.

The CHECKPOINT clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this database level CHECKPOINT clause.

B.1.4.5.3 LOCATION IS directory-spec Specifies the name of the default backing store directory to which all row cache backing files are written. The database system generates a file name automatically (row-cache-name.rdc) for each row cache backing file it creates when the RCS process first starts up. Specify a device name and directory name only, enclosed within single quotation marks. By default, the location is the directory of the database root file.

The LOCATION clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this location which is the default for the database.

B.1.4.5.4 NO LOCATION Removes the location previously specified in a LOCATION IS clause for the database for the row cache backing file. If you specify NO LOCATION, the row cache backing file location becomes the directory of the database root file.

The LOCATION clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this location which is the default for the database.

B.1.4.6 ADD CACHE clause

Creates a new row cache.

B.1.4.6.1 ALLOCATION IS n BLOCK/ALLOCATION IS n BLOCKS Specifies the initial allocation of the row cache backing file (.rdc) to which cached rows are written during a checkpoint operation.

If the ALLOCATION clause is not specified, the default allocation in blocks is approximately 40 percent of the CACHE SIZE for this row cache.

This clause is ignored if the row cache is defined to checkpoint to the database.

B.1.4.6.2 CACHE SIZE IS n ROW/CACHE SIZE IS n ROWS Specifies the number of rows allocated to the row cache. As the row cache fills, rows more recently referenced are retained in the row cache while those not referenced recently are discarded. Adjusting the allocation of the row cache helps to retain important rows in memory. If not specified, the default is 1000 rows.

The product of the `CACHE SIZE` and the `ROW LENGTH` settings determines the amount of memory required for the row cache. (Some additional overhead and rounding up to page boundaries is performed by the database system.) The row cache is shared by all processes attached to the database.

B.1.4.6.3 CHECKPOINT ALL ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO DATABASE Specifies the source and target for checkpoint operations for the row cache. If `ALL ROWS` is specified, then the source records written during each checkpoint operation are both the modified and the unmodified rows in the row cache. If `UPDATED ROWS` is specified, then just the modified rows in the row cache are checkpointed each time.

If the target of the checkpoint operation is `BACKING FILE`, then the `RCS` process writes the source row cache entries to the backing (.rdc) files. The row cache `LOCATION`, `ALLOCATION`, and `EXTENT` clauses are used to create the backing files. Upon recovery from a node failure, the database recovery process is able to re-populate the in-memory row caches from the rows found in the backing files.

If the target is `DATABASE`, then the target rows (only `UPDATED ROWS` is allowed) are written back to the database. The row cache `LOCATION`, `ALLOCATION`, and `EXTENT` clauses are ignored. Upon recovery from a node failure, the database recovery process has no data on the contents of the row cache. Therefore, it does not re-populate the in-memory row caches.

This `CHECKPOINT` clause overrides the database level `CHECKPOINT` clause.

B.1.4.6.4 EXTENT IS n BLOCK/EXTENT IS n BLOCKS Specifies the file extent size for the row cache backing file (.rdc).

If the `EXTENT` clause is not specified, the default number of blocks is `CACHE SIZE * 127` for this cache.

This clause is ignored if the row cache is defined to checkpoint to the database.

B.1.4.6.5 LARGE MEMORY IS ENABLED/LARGE MEMORY IS DISABLED Specifies whether or not large memory is used to manage the row cache. Very large memory (VLM) allows Oracle Rdb to use as much physical memory as is available. It provides access to a large amount of physical memory through small virtual address windows.

Use `LARGE MEMORY IS ENABLED` only when both of the following are true:

- You have enabled row caching.
- You want to cache large amounts of data, but the cache does not fit in the virtual address space.

The default is `DISABLED`. See the Usage Notes for restrictions pertaining to the very large memory (VLM) feature.

B.1.4.6.6 LOCATION IS directory-spec Specifies the name of the default backing store directory to which all row cache backing files are written. The database system generates a file name automatically (row-cache-name.rdc) for each row cache backing file it creates when the RCS process first starts up. Specify a device name and directory name only, enclosed within single quotation marks. By default, the location is the directory of the database root file.

This LOCATION clause overrides a previously specified location at the database level.

This clause is ignored if the row cache is defined to checkpoint to the database.

B.1.4.6.7 NO LOCATION Removes the location previously specified in a LOCATION IS clause for the database for the row cache backing file. If you specify NO LOCATION, the row cache backing file location becomes the directory of the database root file.

This clause is ignored if the row cache is defined to checkpoint to the database.

B.1.4.6.8 NUMBER OF RESERVED ROWS IS n Specifies the maximum number of cache rows that each user can reserve. The default is 20 rows.

The number of reserved rows parameter is also used when searching for available slots in a row cache. The entire row cache is not searched on the initial pass. This parameter is used as the maximum number of rows that are searched for a free slot. If at least one free slot is found, the insert operation can proceed. If no free slots are found in this initial search, Oracle Rdb will continue searching through the cache until it finds a free slot.

B.1.4.6.9 NUMBER OF SWEEP ROWS IS n Specifies the number of modified cache rows that will be written back to the database to make space available in the cache for subsequent transactions to insert rows into the cache. It is recommended that users initially specify the number of sweep rows to be between ten and thirty percent of the total number of rows in the cache. Users should then monitor performance and adjust the number of sweep rows if necessary. The default setting is 3000 rows.

B.1.4.6.10 ROW LENGTH IS n BYTE/ROW LENGTH IS n BYTES Specifies the size of each row allocated to the row cache. Rows are not cached if they are longer than a row cache row. The ROW LENGTH is an aligned longword rounded up to the next multiple of 4 bytes.

If the ROW LENGTH clause is not specified, the default row length is 256 bytes.

B.1.4.6.11 ROW REPLACEMENT IS ENABLED/ROW REPLACEMENT IS DISABLED Specifies whether or not Oracle Rdb replaces rows in the cache. When the ROW REPLACEMENT IS ENABLED clause is used, rows are replaced when the row cache becomes full. When the ROW REPLACEMENT IS DISABLED clause is used, rows are not replaced when the cache is full. The type of row replacement policy depends upon the application requirements for each cache.

The default is ENABLED.

B.1.4.6.12 SHARED MEMORY IS SYSTEM/SHARED MEMORY IS PROCESS

Determines whether cache global sections are created in system space or process space. The default is SHARED MEMORY IS PROCESS.

When you use cache global sections created in the process space, you and other users share physical memory and the OpenVMS Alpha operating system maps a row cache to a private address space for each user. As a result, all users are limited by the free virtual address range and each use a percentage of memory in overhead. If many users are accessing the database, the overhead can be high.

When many users are accessing the database, consider using SHARED MEMORY IS SYSTEM. This gives users more physical memory because they share the system space of memory and there is none of the overhead associated with the process space of memory.

B.1.4.6.13 WINDOW COUNT IS n Specifies the number of virtual address windows used by the LARGE MEMORY clause.

The window is a view into the physical memory used to create the very large memory (VLM) information. Because the VLM size may be larger than that which can be addressed by a 32-bit pointer, you need to view the VLM information through small virtual address windows.

You can specify a positive integer in the range from 10 through 65535. The default is 100 windows.

B.1.4.7 ALTER CACHE row-cache-name

Alters existing row caches.

B.1.4.7.1 row-cache-params For information regarding the row-cache-params, see the descriptions under the ADD CACHE argument described earlier in this arguments list.

B.1.4.7.2 DROP CACHE row-cache-name CASCADE

B.1.4.7.3 DROP CACHE row-cache-name RESTRICT Deletes the specified row cache from the database.

If the mode is RESTRICT, an exception is raised if the row cache is assigned to a storage area.

If the mode is CASCADE, the row cache is removed from all referencing storage areas.

The default is RESTRICT if no mode is specified.

B.2 CREATE DATABASE

B.2.1 Overview

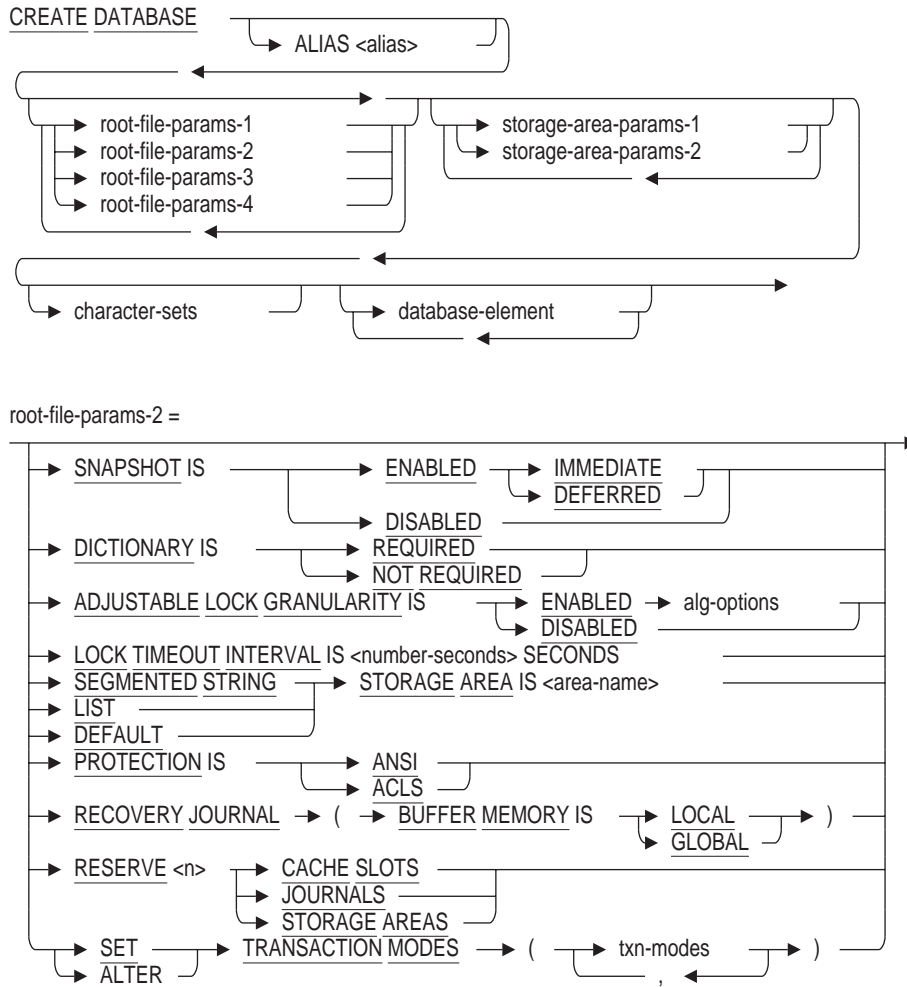
Creates database system files, metadata definitions, and user data that comprise a database. The CREATE DATABASE statement lets you specify in a single SQL statement all data and privilege definitions for a new database. (You can also add definitions to the database later.) For information about ways to ensure good performance and data consistency, see the *Oracle Rdb Guide to Database Performance and Tuning*.

B.2.2 Environment

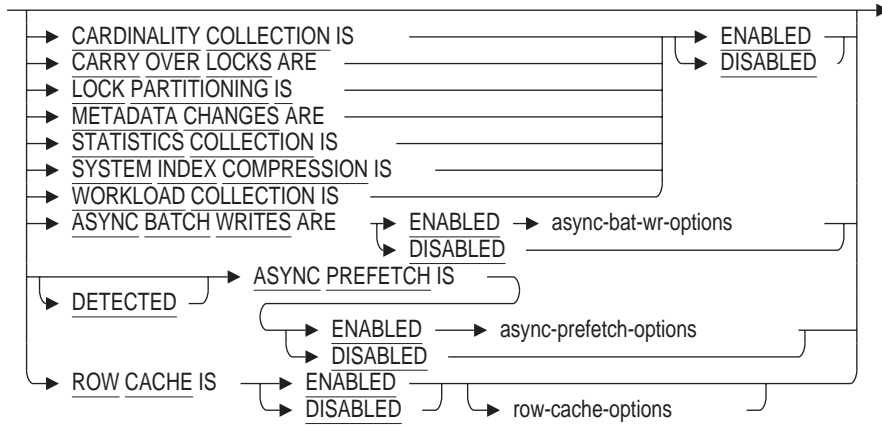
You can use the CREATE DATABASE statement:

- In interactive SQL
- Embedded in host language programs to be precompiled
- As part of a procedure in an SQL module
- In dynamic SQL as a statement to be dynamically executed

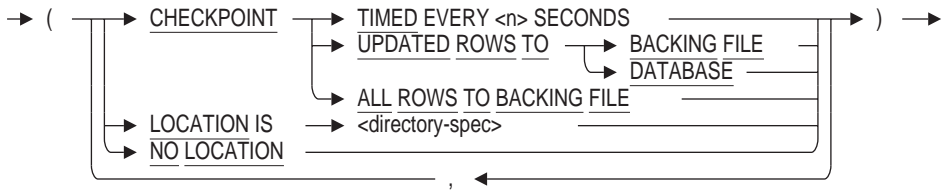
B.2.3 Format



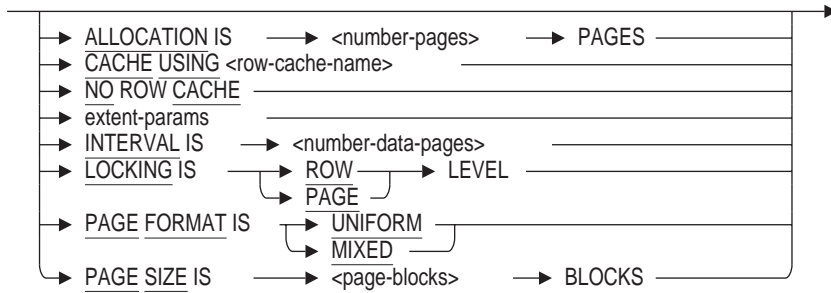
root-file-params-3 =



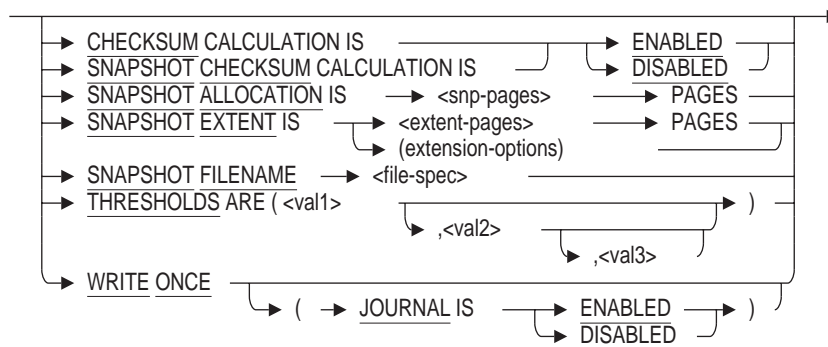
row-cache-options =



storage-area-params-1 =



storage-area-params-2 =



B.2.4 Arguments

B.2.4.1 RECOVERY JOURNAL (BUFFER MEMORY IS {LOCAL | GLOBAL})

The RUJ buffers used by each process are normally allocated in local virtual memory. With the introduction of ROW CACHE, these buffers can now be assigned to a shared global section (GLOBAL memory) so that the recovery process can process this in memory buffer and possibly avoid a disk access.

This buffer memory can be defined a GLOBAL to improve ROW CACHE performance for recovery. If ROW CACHE is DISABLED then buffer memory is always LOCAL.

B.2.4.2 CACHE USING row-cache-name

Assigns the named row cache as the default physical row cache for all storage areas in the database. All rows stored in each storage area, whether they consist of table data, segmented string data, or special rows such as index nodes, are cached.

You must create the row cache before terminating the CREATE DATABASE statement. For example:

```

SQL> CREATE DATABASE FILENAME test_db
cont> ROW CACHE IS ENABLED
cont> CACHE USING test1
cont> CREATE CACHE test1
cont>   CACHE SIZE IS 100 ROWS
cont> CREATE STORAGE AREA areal;

```

If you do not specify the CACHE USING clause or the NO ROW CACHE clause, NO ROW CACHE is the default for the database.

You can override the database default row cache by either specifying the CACHE USING clause after the CREATE STORAGE AREA clause or by later altering the database and storage area to assign a new row cache. Only one physical area row cache is allowed for each storage area.

You can have multiple row caches containing rows for a single storage area by defining logical area row caches, where the row cache name matches the name of a table or index.

B.2.4.2.1 NO ROW CACHE Specifies that the database default is not to assign a row cache to all storage areas in the database. You cannot specify the NO ROW CACHE clause if you specify the CACHE USING clause.

Alter the storage area and name a row cache to override the database default. Only one row cache is allowed for each storage area.

If you do not specify the CACHE USING clause or the NO ROW CACHE clause, NO ROW CACHE is the default for the database.

B.2.4.3 RESERVE n CACHE SLOTS

Specifies the number of row caches for which slots are reserved in the database.

You can use the RESERVE CACHE SLOTS clause to reserve slots in the database root file for future use by the ADD CACHE clause. Row caches can be added only if there are row cache slots available. Slots become available after a DROP CACHE clause or a RESERVE CACHE SLOTS clause.

The number of reserved slots for row caches cannot be decreased once the RESERVE clause is issued. If you reserve 10 slots and later reserve 5 slots, you have a total of 15 reserved slots for row caches.

Reserving row cache slots is an offline operation (requiring exclusive database access). See the Section B.1 for more information about row caches.

B.2.4.4 ROW CACHE IS ENABLED/ROW CACHE IS DISABLED

Specifies whether or not you want Oracle Rdb to enable the row caching feature.

When a database is created or is converted from a previous version of Oracle Rdb without specifying row cache support, the default is ROW CACHE IS DISABLED. Enabling row cache support does not affect database operations until a row cache area is created and assigned to one or more storage areas.

When the row caching feature is disabled, all previously created and assigned row caches remain in existence for future use when the row caching feature is enabled.

B.2.4.4.1 CHECKPOINT TIMED EVERY N SECONDS Specifies the frequency with which the RCS process checkpoints the contents of the row caches back to disk. **The RCS process does not use the checkpoint frequency options of the FAST COMMIT clause.**

The frequency of RCS checkpointing is important in determining how much of an AIJ file must be read during a REDO operation following a node failure. It also affects the frequency that marked records get flushed back to the database, for those row caches that checkpoint to the database. The default is every 15 minutes (900 seconds).

B.2.4.4.2 CHECKPOINT ALL ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO DATABASE Specifies the default source and target for checkpoint operations for all row caches. If ALL ROWS is specified, then the source records written during each checkpoint operation are both the modified and the unmodified rows in a row cache. If UPDATED ROWS is specified, then just the modified rows in a row cache are checkpointed each time.

If the target of the checkpoint operation is BACKING FILE, then the RCS process writes the source row cache entries to the backing (.rdc) files. The row cache LOCATION, ALLOCATION, and EXTENT clauses are used to create the backing files. Upon recovery from a node failure, the database recovery process is

able to re-populate the in-memory row caches from the rows found in the backing files.

If the target is DATABASE, then the target rows (only UPDATED ROWS is allowed) are written back to the database. The row cache LOCATION, ALLOCATION, and EXTENT clauses are ignored. Upon recovery from a node failure, the database recovery process has no data on the contents of the row cache. Therefore, it does not re-populate the in-memory row caches.

The CHECKPOINT clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this database level CHECKPOINT clause.

B.2.4.4.3 LOCATION IS directory-spec Specifies the name of the default backing store directory to which all row cache backing files are written. The database system generates a file name automatically (row-cache-name.rdc) for each row cache backing file it creates when the RCS process first starts up. Specify a device name and directory name only, enclosed within single quotation marks. By default, the location is the directory of the database root file.

The LOCATION clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this location which is the default for the database.

B.2.4.4.4 NO LOCATION Removes the location previously specified in a LOCATION IS clause for the database for the row cache backing file. If you specify NO LOCATION, the row cache backing file location becomes the directory of the database root file.

The LOCATION clause of the CREATE CACHE, ADD CACHE, or ALTER CACHE clause overrides this location which is the default for the database.

B.3 CREATE CACHE Clause

Creates a row cache area that allows frequently referenced rows to remain in memory even when the associated page has been transferred back to disk. This saves in memory usage because only the more recently referenced rows are cached versus caching the entire buffer.

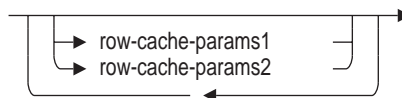
See the Section B.1 and the Section B.2 for more information regarding the row cache areas.

B.3.1 Environment

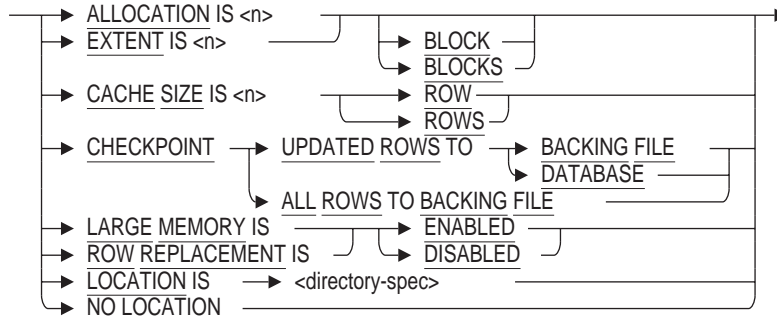
You can use the CREATE CACHE clause only within a CREATE DATABASE or IMPORT statement.

B.3.2 Format

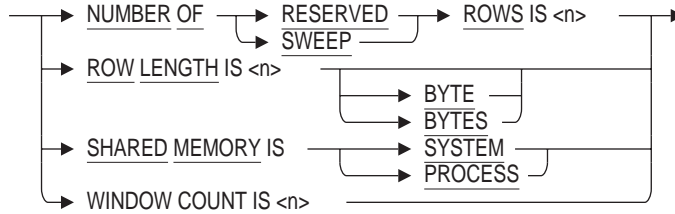
CREATE CACHE <row-cache-name>



row-cache-params1 =



row-cache-params2 =



B.3.3 Arguments

B.3.3.0.1 CACHE row-cache-name Creates a row cache.

B.3.3.0.2 ALLOCATION IS n BLOCK/ALLOCATION IS n BLOCKS Specifies the initial allocation of the row cache file (.rdc) to which cached rows are written during a checkpoint operation.

If the ALLOCATION clause is not specified, the default allocation in blocks is approximately 40 percent of the CACHE SIZE for this cache.

This clause is ignored if the row cache is defined to checkpoint to the database.

B.3.3.0.3 EXTENT IS n BLOCK/EXTENT IS n BLOCKS Specifies the file extent size for the row cache backing file (.rdc).

If the EXTENT clause is not specified, the default number of blocks is CACHE SIZE * 127 for this cache.

This clause is ignored if the row cache is defined to checkpoint to the database.

B.3.3.0.4 CACHE SIZE IS n ROW/CACHE SIZE IS n ROWS Specifies the number of rows allocated to the row cache. As the row cache fills, rows more recently referenced are retained in the row cache while those not referenced recently are discarded. Adjusting the allocation of the row cache helps to retain important rows in memory. If not specified, the default is 1000 rows.

The product of the CACHE SIZE and the ROW LENGTH settings determines the amount of memory required for the row cache. (Some additional overhead and rounding up to page boundaries is performed by the database system.) The row cache is shared by all processes attached to the database.

B.3.3.0.5 CHECKPOINT ALL ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO BACKING FILE/ CHECKPOINT UPDATED ROWS TO DATABASE Specifies the source and target for checkpoint operations for the row cache. If ALL ROWS is specified, then the source records written during each checkpoint operation are both the modified and the unmodified rows in the row cache. If UPDATED ROWS is specified, then just the modified rows in the row cache are checkpointed each time.

If the target of the checkpoint operation is BACKING FILE, then the RCS process writes the source row cache entries to the backing (.rdc) files. The row cache LOCATION, ALLOCATION, and EXTENT clauses are used to create the backing files. Upon recovery from a node failure, the database recovery process is able to re-populate the in-memory row caches from the rows found in the backing files.

If the target is DATABASE, then the target rows (only UPDATED ROWS is allowed) are written back to the database. The row cache LOCATION, ALLOCATION, and EXTENT clauses are ignored. Upon recovery from a node failure, the database recovery process has no data on the contents of the row cache. Therefore, it does not re-populate the in-memory row caches.

This CHECKPOINT clause overrides the database level CHECKPOINT clause.

B.3.3.0.6 LARGE MEMORY IS ENABLED/LARGE MEMORY IS DISABLED Specifies whether or not large memory is used to manage the row cache. Very large memory (VLM) allows Oracle Rdb to use as much physical memory as is available. It provides access to a large amount of physical memory through small virtual address windows.

Use LARGE MEMORY IS ENABLED only when both of the following are true:

- You have enabled row caching.
- You want to cache large amounts of data, but the cache does not fit in the virtual address space.

The default is DISABLED.

See the Usage Notes for restrictions pertaining to the very large memory (VLM) feature.

B.3.3.0.7 ROW REPLACEMENT IS ENABLED/ROW REPLACEMENT IS DISABLED Specifies whether or not Oracle Rdb replaces rows in the cache. When the ROW REPLACEMENT IS ENABLED clause is used, rows are replaced when the row cache becomes full. When the ROW REPLACEMENT IS DISABLED clause is used, rows are not replaced when the cache is full. The type of row replacement policy depends upon the application requirements for each cache.

The default is ENABLED.

B.3.3.0.8 LOCATION IS directory-spec Specifies the name of the default backing store directory to which all row cache backing files are written. The database system generates a file name automatically (row-cache-name.rdc) for each row cache backing file it creates when the RCS process first starts up. Specify a device name and directory name only, enclosed within single quotation marks. By default, the location is the directory of the database root file.

This LOCATION clause overrides a previously specified location at the database level.

This clause is ignored if the row cache is defined to checkpoint to the database.

B.3.3.0.9 NO LOCATION Removes the location previously specified in a LOCATION IS clause for the database for the row cache backing file. If you specify NO LOCATION, the row cache backing file location becomes the directory of the database root file.

This clause is ignored if the row cache is defined to checkpoint to the database.

B.3.3.0.10 NUMBER OF RESERVED ROWS IS n Specifies the maximum number of cache rows that each user can reserve. The default is 20 rows.

The number of reserved rows parameter is also used when searching for available slots in a row cache. The entire row cache is not searched on the initial pass. This parameter is used as the maximum number of rows that are searched for a free slot. If at least one free slot is found, the insert operation can proceed. If no free slots are found in this initial search, Oracle Rdb will continue searching through the cache until it finds a free slot.

B.3.3.0.11 NUMBER OF SWEEP ROWS IS n Specifies the number of modified cache rows that will be written back to the database to make space available in the cache for subsequent transactions to insert rows into the cache. It is recommended that users initially specify the number of sweep rows to be between ten and thirty percent of the total number of rows in the cache. Users should then monitor performance and adjust the number of sweep rows if necessary. The default setting is 3000 rows.

B.3.3.0.12 ROW LENGTH IS n BYTE/ROW LENGTH IS n BYTES Specifies the size of each row allocated to the row cache. Rows are not cached if they are longer than a row cache row. The ROW LENGTH is an aligned longword rounded up to the next multiple of 4 bytes.

If the ROW LENGTH clause is not specified, the default row length is 256 bytes. The maximum row length in a row cache area is 65535 bytes.

If the ROW LENGTH clause is not specified, the default row length is 256 bytes.

B.3.3.0.13 SHARED MEMORY IS SYSTEM/SHARED MEMORY IS PROCESS Determines whether cache global sections are created in system space or process space. The default is SHARED MEMORY IS PROCESS.

When you use cache global sections created in the process space, you and other users share physical memory and the OpenVMS Alpha operating system maps a row cache to a private address space for each user. As a result, all users are limited by the free virtual address range and each use a percentage of memory in overhead. If many users are accessing the database, the overhead can be high.

When many users are accessing the database, consider using SHARED MEMORY IS SYSTEM. This gives users more physical memory because they share the system space of memory and there is none of the overhead associated with the process space of memory.

B.3.3.0.14 WINDOW COUNT IS n Specifies the number of virtual address windows used by the LARGE MEMORY clause.

The window is a view into the physical memory used to create the very large memory (VLM) information. Because the VLM size may be larger than that which can be addressed by a 32-bit pointer, you need to view the VLM information through small virtual address windows.

You can specify a positive integer in the range from 10 through 65535. The default is 100 windows.

B.3.4 Usage Notes

- If the name of the row cache is the same as any logical area (for example a table name, index name, storage map name, RDB\$SEGMENTED_STRINGS, RDB\$SYSTEM_RECORD, and so forth), then this is a logical area cache and the named logical area is cached automatically. Otherwise, a storage area needs to be associated with the cache.
- The CREATE CACHE clause does not assign the row cache to a storage area. You must use the CACHE USING clause with the CREATE STORAGE AREA clause of the CREATE DATABASE statement or the CACHE USING clause with the ADD STORAGE AREA or ALTER STORAGE AREA clauses of the ALTER DATABASE statement.
- The product of the CACHE SIZE and the ROW LENGTH settings determines the amount of memory required for the row cache (some additional overhead and rounding up to page boundaries is performed by the database system).
- The row cache is shared by all processes attached to the database on any one node.
- The following are requirements when using the row caching feature:
 - After-image journaling must be enabled
 - Fast commit must be enabled
 - Number of cluster nodes must equal 1
- Use the SHOW CACHE statement to view information about a cache.

Release Notes Relating to the Row Cache Feature

This section describes software errors that were fixed by Oracle Rdb7 Release 7.0.1.5 and 7.0.1.6 relating specifically to the row cache feature.

C.1 Software Errors Fixed That Apply to All Interfaces

C.1.1 RCS Maximum Log File Size Control Logical

In prior versions of Oracle Rdb7, the Row Cache Server (RCS) process log file (enabled via the `RDM$BIND_RCS_LOG_FILE` logical name) would continue to grow until the database was shut down. This would be a significant problem because when the disk containing the log file would become full, the RCS process could fail.

The RCS process log file maximum size can now be controlled with the system logical name `RDM$BIND_RCS_LOG_REOPEN_SIZE`. This logical, when defined before the database is opened, limits the allocated size of the RCS log file. When the log file allocation reaches the specified number of disk blocks, the current log file will be closed and a new log file opened. Older log files can be archived or purged as needed.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.5.

C.1.2 New `RMU /SET ROW_CACHE [/ENABLE | /DISABLE]` Command

A new `RMU /SET` command "`ROW_CACHE`" has been added to allow the database Row Cache feature to be enabled or disabled without requiring that the database be opened. This command requires exclusive database access (the database can not be open or be accessed by other users).

Valid qualifiers for the "`RMU /SET ROW_CACHE`" command are:

- `/ENABLE` to enable row caching
- `/DISABLE` to disable row caching
- `/LOG` to display a log message at the completion of the `RMU /SET` operation

The `/ENABLE` and `/DISABLE` qualifiers are mutually exclusive.

This command has been added to Oracle Rdb7 Release 7.0.1.5.

C.1.3 RCS Clearing "GRIC" Reference Counts

When the Oracle Rdb7 Row Cache feature is enabled, the Row Cache Server (RCS) process will attempt to clear the reference count field in a data structure called a GRIC. The reference count will be cleared periodically based on the number of DBR (Database Recovery) processes run. If enough DBR processes have run, a Row Cache "sweep" request can trigger the reference count clearing.

When a process that uses a row cache abnormally terminates (via STOP/ID, for example), it can leave references in the cache that would prevent rows in the cache from being removed. This can cause the cache to become full of rows that are not really referenced by any process though they appear to be referenced due to an elevated reference count.

A Row Cache "sweep" request to the RCS process indicates that a cache is "full" and there is no more room to insert new rows into the cache. When the RCS process receives the sweep request, it will see if a number of DBRs have run since the last sweep. If enough DBRs have run (the default is 25 DBRs since the last sweep for the cache), the RCS will initiate a "Release GRICs" operation.

This operation can have a minor performance impact to users of the cache and can also delay the RCS from performing other operations. This is why it is a periodic event.

The system logical name `RDM$BIND_RCS_CLEAR_GRICS_DBR_CNT` can be used to control the number of DBRs that must elapse before the RCS will initiate clearing of the GRIC reference counts. The maximum value of the logical name is "100000". The default value (if the logical name is not defined) is "25". Defining the logical name with a value of "0" disables clearing the reference counts.

For most systems, the default value is adequate. However, systems with very frequent database recoveries may need a high value of the logical name to reduce the frequency that the reference counts are cleared. The RCS process log file can be used to determine how often the reference counts are cleared.

This new logical name has been included in Oracle Rdb7 Release 7.0.1.5.

C.1.4 Row Cache RDC File Name Change

In the previous release of Oracle Rdb7, the Row Cache backing store file used a file type of ".RDC". This behavior caused a file name conflict when a database was replicated either with the RMU/COPY command or when using the "Hot Standby" feature.

This conflict has been resolved in Oracle Rdb7 Release 7.0.1.5. The Row Cache backing store file type has been extended to include the root file device name and file ID in a BASE32 format (where valid characters are 0 to 9 and A to W).

For example, a row cache backing store file name may now have a format similar to the following:

```
EMPIDX_10_0.RDC_0C1H85848NO00063228L;1
```

In this example, the value "0C1H85848NO00063228L" represents the device name and file ID of the root file for the database. The file type is always prefixed with ".RDC_". All Row Cache backing store files for a database have this same exact file type. Another database using the same location for backing store files would use a different file type (perhaps ".RDC_4D87HD234FSD0063228L").

To associate a database with a Row Cache backing store file, the "RMU /DUMP /CACHE_FILE" command can be used to display the Row Cache backing store file header when the full name of the database root file is stored.

Because existing Row Cache backing store files have a file type of ".RDC", if you use the `RDM$BIND_RCS_KEEP_BACKING_FILES` logical to keep existing backing store files from being deleted when a database is closed, you should deassign the logical prior to closing the database(s) in preparation for installing Oracle Rdb7 Release 7.0.1.5. This will allow existing ".RDC" files to be deleted properly.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.6. The erased row is now correctly detected and the row that is too large for the cache is now returned from disk.

C.1.7 Overriding RCS Checkpoint Timer Interval

In the prior versions of Oracle Rdb7, the Row Cache Server (RCS) process' checkpoint timer interval could be overridden by the system logical "RDMSBIND_CKPT_TIME". This is the logical that allows the fast commit checkpoint timer interval to be overridden. Using the same logical for the RCS checkpoint timer was confusing and error prone.

Beginning with Oracle Rdb7 Release 7.0.1.6, the RCS process' checkpoint timer interval can be overridden with a new system logical name, "RDMSBIND_RCS_CKPT_TIME".

If neither this logical nor the "ROW CACHE IS ENABLED (CHECKPOINT TIMED EVERY n SECONDS)" database clause is specified, then the RCS process will use the "RDMSBIND_CKPT_TIME" logical name or its associated dashboard value.

If RCS still has a zero checkpoint timer interval, then it will default to a fixed 15 minute interval.

C.1.8 Refresh RCS Metadata Information

In the prior versions of Oracle Rdb7, the Row Cache Server (RCS) process would maintain its metadata structures across checkpoint and sweep requests. While the RCS process was active, however, Oracle Rdb7 would allow tables, indices, and storage areas to be dropped and recreated. In these situations, it was possible for the RCS process to not notice the metadata changes and use the original metadata to write modified records from the row caches back to the original database storage areas. This would result in database corruptions and bugcheck dumps.

In Oracle Rdb7 Release 7.0.1.6, the RCS now recognizes that if it is not holding the corresponding logical or physical area locks, its metadata may be obsolete. When this occurs, the RCS process refreshes its metadata structures from the AIP and root file information.

C.1.9 RCS ACCVIO When Checkpointing All Row Caches to Database

Beginning with Release 7.0.1.5 of Oracle Rdb7, the Row Cache Server (RCS) process would inadvertently access violate after completing its final checkpoint to the database as part of a database shutdown operation. It was access violating while trying to clean up a data structure that had not been allocated.

This problem does not corrupt the database. Simply reopen the database and database access will be fine until the database is closed again, whereupon this problem will be hit again. A workaround to this problem is to have at least one row cache checkpoint to a backing file.

This problem has been corrected in Oracle Rdb7 Release 7.0.1.6.

Known Problems and Restrictions Relating to the Row Cache Feature

This section describes known problems and restrictions relating to the row cache feature and includes workarounds where appropriate. Unless otherwise noted, all notes apply to all platforms.

D.1 Known Problems and Restrictions

D.1.1 RMU Online Verification Operations and Row Cache

When using row caches, some RMU online verification operations may report errors in the database structure and may not be generally reliable in all verifications. These errors may be due to RMU validating the on-disk database structure and not the actual logical database structure including the row cache contents.

For example, one of the verifications that is performed by RMU/VERIFY is to ensure that system records in mixed format areas have a “system record” record ID. However, when a physical row cache is being used, the row on the database page may be marked as “reserved by record cache” because the row has been modified in the row cache but has not yet been flushed to disk.

In the following example, the database ID of **00002011** refers to the “reserved by record cache” record type and **00002001** refers to the system record type:

```
$ RMU/VERIFY/ONLINE DKA0:[DB]MYDB.RDB:1
%RMU-E-PAGSYSREC, area INDEX_MIXED_AREA, page 3
      system record contains an invalid database ID
      expected: 00002001 (hex), found: 00002011 (hex)
```

D.1.2 Limitation: Online RMU /VERIFY and Row Cache

Performing online RMU /VERIFY operations on a database with the Row Cache feature enabled may report errors even though there is actually no problem. RMU /VERIFY is not fully integrated with the Row Cache feature in this release. Because of this, if there is database modification activity occurring while the verify is running, misleading error messages may be displayed.

If possible, limit online RMU /VERIFY operations to times when the database is not being actively modified or perform an offline database verification.

This problem will be corrected in a future Oracle Rdb release.

D.1.3 Adding Row Caches Requires Exclusive Database Access

Adding a row cache with the ALTER DATABASE ADD CACHE command now requires exclusive database access.

Previously, it was possible for a new row cache to be added online. This new cache would be seen by users attaching to the database after the cache was created, but users that were already attached to the database would not be able to access the cache and would return results from the database without referencing the cache. This situation resulted in database corruption.

D.1.4 Conflicts When Caching Metadata and Executing Certain SQL Database Operations

When caching metadata, you will experience conflicts when executing database operations through SQL that require exclusive database access. For example, adding new row caches or dropping existing ones requires exclusive database access. When the SQL command is parsed, the Oracle Rdb system tables are queried. This access to the system tables creates the row caches and causes the RCS process to come up to manage those row caches. As a result, the database now has another “user”, the RCS process. This causes the exclusive database operation to fail.

To resolve this, you must first turn off row caching temporarily using the RMU Set command specifying the Row_Cache and Disabled qualifiers. Then, perform the SQL operation that requires exclusive database access. Finally, re-enable row caching using the RMU Set command with the Row_Cache and Enabled qualifiers.

E

Logical Names Relating to the Row Cache Feature

This section describes logical names relating specifically to the row cache feature and explains when and how to use them. Note that the fields following the logical name list the table name in which the logical must be defined and the value of the logical with defaults given where applicable.

E.1 RDM\$BIND_CKPT_FILE_SIZE

RDM\$BIND_CKPT_FILE_SIZE LNM\$FILE_DEV INTEGER

This logical represents the percentage of the row cache size that you want the backing file allocation to be. Applied to all backing files. This overrides the backing file's allocation specified in the CREATE/ADD CACHE definition.

E.2 RDM\$BIND_CKPT_TIME

RDM\$BIND_CKPT_TIME LNM\$FILE_DEV INTEGER (Default=0)

This logical represents the frequency of RCS checkpoint. It overrides the "Alter database row cache is enabled (checkpoint timed every N seconds)" value.

E.3 RDM\$BIND_DBR_UPDATE_RCACHE

RDM\$BIND_DBR_UPDATE_RCACHE LNM\$SYSTEM_TABLE 0 or 1(Default)

If the logical is set to 0, during recovery from node failure, don't repopulate in-memory row caches from their backing files (only recover the database). If the logical is set to 1 (the default), during recovery from node failure, repopulate in-memory row caches from backing files and from REDO operations.

E.4 RDM\$BIND_RCACHE_INSERT_ENABLED

RDM\$BIND_RCACHE_INSERT_ENABLED LNM\$FILE_DEV 0 or 1(Default)

This is a process logical. If the logical is set to 0, this process cannot insert any rows into the row caches; this process can only use what is already there. If the logical is set to 1 (the default), the process can insert new rows into the row cache, if they fit.

E.5 RDM\$BIND_RCACHE_LATCH_SPIN_COUNT

RDM\$BIND_RCACHE_LATCH_SPIN_COUNT LNM\$FILE_DEV INTEGER (Default=1024)

This logical represents how many iterations to retry getting the row cache latch before hibernating. This consumes CPU but can acquire the latch faster. Set in 1000s.

E.6 RDM\$BIND_RCACHE_RCRL_COUNT

RDM\$BIND_RCACHE_RCRL_COUNT LNM\$FILE_DEV INTEGER (Default=0)

This logical represents the number of rows to reserve when acquiring empty slots in a row cache. This overrides the “NUMBER OF RESERVE ROWS IS N” clause.

E.7 RDM\$BIND_RCS_BATCH_COUNT

RDM\$BIND_RCS_BATCH_COUNT LNM\$SYSTEM_TABLE INTEGER (Default=3000)

This logical represents the number of rows RCS attempts to write out at a time during the course of a checkpoint or sweep.

E.8 RDM\$BIND_RCS_CARRYOVER_ENABLED

RDM\$BIND_RCS_CARRYOVER_ENABLED LNM\$SYSTEM_TABLE 0 or 1(Default)

If the logical is set to 0, RCS doesn't honor carryover locks for logical/physical areas. It continues to hold them (good for RCS performance, but prevents exclusive access to these logical/physical areas). If the logical is set to 1 (the default), RCS honors carryover locks and gives up logical/physical area locks it is holding that it is not using but that simply remain from a prior operation.

E.9 RDM\$BIND_RCS_CKPT_COLD_ONLY

RDM\$BIND_RCS_CKPT_COLD_ONLY LNM\$SYSTEM_TABLE 0(Default) or 1

If the logical is set to 0 (the default), checkpoint/sweep all marked records in a row cache. If the logical is set to 1, only checkpoint records marked before the PRIOR ckpt interval (only checkpoint the older/colder data, but this also keeps the RCS ckpt farther behind causing more AIJ to read during REDO).

E.10 RDM\$BIND_RCS_CKPT_BUFFER_CNT

RDM\$BIND_RCS_CKPT_BUFFER_CNT LNM\$SYSTEM_TABLE INTEGER (Default=15)

This logical represents the number of buffers to use to write records to backing files during checkpoints.

E.11 RDM\$BIND_RCS_CKPT_TIME

RDM\$BIND_RCS_CKPT_TIME LNM\$SYSTEM_TABLE INTEGER (Default=0)

This logical overrides the RCS process' checkpoint timer interval. This logical was added in Release 7.0.1.6. If neither this logical nor the “ROW CACHE IS ENABLED (CHECKPOINT TIMED EVERY n SECONDS)” database clause is specified, then the RCS process will use the “RDM\$BIND_CKPT_TIME” logical name or its associated dashboard value. If RCS still has a zero checkpoint timer interval, then it will default to a fixed 15 minute interval.

E.12 RDM\$BIND_RCS_CLEAR_GRICS_DBR_CNT

RDM\$BIND_RCS_CLEAR_GRICS_DBR_CNT LNM\$SYSTEM_TABLE INTEGER (Default=25)

This logical represents the frequency (based on the number of DBR processes that run) with which the RCS will attempt to release references in the cache left by abnormally terminated processes. For each sweep request for a cache, if at least this number of DBR processes have run since the last sweep for the cache, the RCS will initiate a "Release GRICS" operation. This operation can have a minor performance impact to users of the cache and can also delay the RCS from

performing other operations. This is why it is a periodic event. The maximum value of the logical is 100000. The default value is 25. Defining the logical name with a value of 0 will disable the clearing of reference counts.

E.13 RDM\$BIND_RCS_CREATION_IMMEDIATE

RDM\$BIND_RCS_CREATION_IMMEDIATE LNM\$SYSTEM_TABLE 0(Default) or 1

If the logical is set to 0 (the default), for automatic open database, create RCS process on first reference to a row cache. If the logical is set to 1, for automatic open database, create RCS process on initial attach. If the logical is set to 1, for manual open database, RCS is started immediately.

E.14 RDM\$BIND_RCS_KEEP_BACKING_FILES

RDM\$BIND_RCS_KEEP_BACKING_FILES LNM\$SYSTEM_TABLE 0(Default) or 1

If the logical is set to 0 (the default), the RCS creates/deletes backing files on each startup/shutdown. If the logical is set to 1, the RCS retains backing files on shutdown and reuses them on startup.

E.15 RDM\$BIND_RCS_LOG_FILE

RDM\$BIND_RCS_LOG_FILE LNM\$SYSTEM_TABLE File Name

This logical specifies the location and name of the optional RCS process log file. If the logical is not defined, no RCS logging is done. It is recommended that logging be turned on. If a location is not specified along with the file name, the log file is created in the same location as the database root file.

E.16 RDM\$BIND_RCS_LOG_HEADER

RDM\$BIND_RCS_LOG_HEADER LNM\$SYSTEM_TABLE 0 or 1(Default)

If the logical is set to 0, don't insert header sections in RCS log file. If the logical is set to 1 (the default), insert normal header sections into the RCS log file.

E.17 RDM\$BIND_RCS_LOG_REOPEN_SIZE

RDM\$BIND_RCS_LOG_REOPEN_SIZE LNM\$SYSTEM_TABLE INTEGER (Default=0)

This logical represents the maximum block size of the RCS log file before the RCS opens a new log file.

E.18 RDM\$BIND_RCS_LOG_REOPEN_SECS

RDM\$BIND_RCS_LOG_REOPEN_SECS LNM\$SYSTEM_TABLE INTEGER (Default=0)

This logical, when defined before the database is opened, causes the RCS log file to be reopened after every 'n' seconds as specified by the value of the logical name. If the value of the logical is 0 or it is not defined, then the RCS Log file is not reopened based on time. The maximum value allowed is 31449600 (which is one year noted in seconds).

E.19 RDM\$BIND_RCS_PRIORITY

RDM\$BIND_RCS_PRIORITY LNM\$SYSTEM_TABLE INTEGER

This logical represents the base priority of the RCS process.

E.20 RDM\$BIND_RCS_SWEEP_COUNT

RDM\$BIND_RCS_SWEEP_COUNT LNM\$SYSTEM_TABLE INTEGER

This logical represents the number of rows to sweep. It overrides the "NUMBER OF SWEEP ROWS IS N" clause.

E.21 RDM\$BIND_RCS_VALIDATE_SECS

RDM\$BIND_RCS_VALIDATE_SECS LNM\$SYSTEM_TABLE INTEGER

This logical defines the number of seconds between each cache validation pass. A value in the range of 300 (5 minutes) to 86400 (24 hours) is suggested. A value of 0 disables the cache validations. Once initiated, the interval can be re-set by changing the logical name definition; the logical is translated at each validation.

E.22 RDM\$BIND_RUJ_GLOBAL_SECTION_ENABLED

RDM\$BIND_RUJ_GLOBAL_SECTION_ENABLED LNM\$SYSTEM_TABLE 0 or 1
(Default=1 if row cache enabled)
(Default=0 if row cache disabled)

If the logical is set to 0, don't place RUJ I/O buffers in global section so DBR can see them. If the logical is set to 1, place RUJ I/O buffers in global section so DBR can see them.