

# **Oracle® Rdb for OpenVMS**

# Table of Contents

<b><u>Oracle® Rdb for OpenVMS</u></b> .....	1
<b><u>Release Notes</u></b> .....	2
<b><u>December 2005</u></b> .....	3
<b><u>Contents</u></b> .....	4
<b><u>Preface</u></b> .....	5
<b><u>Purpose of This Manual</u></b> .....	6
<b><u>Intended Audience</u></b> .....	7
<b><u>Document Structure</u></b> .....	8
<b><u>Chapter 1 Installing Oracle Rdb Release 7.1.4.3</u></b> .....	9
<b><u>1.1 Alpha EV7 Processor Support</u></b> .....	10
<b><u>1.2 Oracle Rdb V7.1 Version Numbering Enhancement</u></b> .....	11
<b><u>1.3 Requirements</u></b> .....	12
<b><u>1.4 Invoking VMSINSTAL</u></b> .....	13
<b><u>1.5 Stopping the Installation</u></b> .....	14
<b><u>1.6 After Installing Oracle Rdb</u></b> .....	15
<b><u>1.7 Spurious SYSVERDIF Message During Installation</u></b> .....	16
<b><u>1.8 Patches for OpenVMS V7.3-1</u></b> .....	17
<b><u>1.9 Oracle Rdb Release 7.1.4.3.1 Optimized for Alpha EV56 (21164A Processor Chip) and Later Platforms</u></b> .....	18
<b><u>1.9.1 AlphaServer 4000 EV56 299Mhz Not Supported by Oracle Rdb Release Optimized for Alpha EV56 Processor</u></b> .....	19
<b><u>1.10 Maximum OpenVMS Version Check Added</u></b> .....	20
<b><u>1.11 VMS\$MEM RESIDENT USER Rights Identifier Required</u></b> .....	21
<b><u>Chapter 2 Software Errors Fixed in Oracle Rdb Release 7.1.4.3</u></b> .....	22
<b><u>2.1 Software Errors Fixed That Apply to All Interfaces</u></b> .....	23
<b><u>2.1.1 Various Bugchecks Due to Memory Corruption</u></b> .....	23
<b><u>2.1.2 Reserved Sequence Slots Not Properly Initialized</u></b> .....	23

# Table of Contents

<b><u>2.1 Software Errors Fixed That Apply to All Interfaces</u></b>	<b>24</b>
<u>2.1.3 User Attach Stalls Waiting for ALS Startup</u>	24
<b><u>2.2 SQL Errors Fixed</u></b>	<b>26</b>
<u>2.2.1 CREATE TABLE ... LIKE May Create Invalid NOT NULL Constraints</u>	26
<u>2.2.2 STDDEV and VARIANCE Generate Wrong Values in GROUP BY Queries</u>	26
<u>2.2.3 SQL Module Language /PROTOTYPES Now Generates INT64 for BIGINT Parameters</u>	27
<u>2.2.4 IMPORT DATABASE Loses SECURITY CHECKING Attributes</u>	28
<b><u>2.3 RMU Errors Fixed</u></b>	<b>29</b>
<u>2.3.1 RMU/BACKUP of Database With Very Large Storage Area Count</u>	29
<u>2.3.2 RMU Backup and Restore Problems With Many Storage Areas</u>	29
<u>2.3.3 Corrections to RMU Extract</u>	30
<b><u>2.4 Row Cache Errors Fixed</u></b>	<b>32</b>
<u>2.4.1 RCS and User Memory Leak With Snapshots in Row Cache</u>	32
<b><u>2.5 RMU Show Statistics Errors Fixed</u></b>	<b>33</b>
<u>2.5.1 Excessive Erases Done for Fragmented Rows</u>	33
<u>2.5.2 Some RMU/SHOW STATISTICS Screens Not Cleared Properly</u>	34
<b><u>Chapter 3 Software Errors Fixed in Oracle Rdb Release 7.1.4.2</u></b>	<b>35</b>
<b><u>3.1 Software Errors Fixed That Apply to All Interfaces</u></b>	<b>36</b>
<u>3.1.1 Bugcheck in PSII2REMOVEBOTTOM</u>	36
<u>3.1.2 Logical RDM\$BIND RW TX CHECKPOINT ADVANCE Removed</u>	36
<u>3.1.3 Wrong Result from Constant View Column</u>	36
<u>3.1.4 Monitor Fails with VASFULL Error</u>	38
<u>3.1.5 Wrong Result from UNION View Query with NOT STARTING WITH Clause</u>	39
<u>3.1.6 Problem with Partitioned Indexes and Descending Segments with Starting With and Like</u>	41
<u>3.1.7 AIJ Server Process Not Always Run Under RDM\$AIJ Account</u>	43
<u>3.1.8 ALS Would Not Stay Stopped</u>	43
<u>3.1.9 Bugchecks at PSIHASHBKT\$INSERT INTO SYS + 00000068</u>	44
<u>3.1.10 Bugchecks or Corruption in Indexes of TYPE IS SORTED RANKED</u>	44
<u>3.1.11 Unexpected Bugcheck when Formatting Illegal Date/Time Value</u>	45
<u>3.1.12 Wrong Result by Query with Constant Column Defined in a View</u>	46
<u>3.1.13 Various Errors Using Read Only Areas with Global Buffers</u>	49
<u>3.1.14 Many Processes Stalling with "hibernating on AIJ submission"</u>	49
<u>3.1.15 DBR Bugchecks at PIO\$BIND + 000003B8</u>	50
<u>3.1.16 Recovered Database Missing Updates</u>	51
<u>3.1.17 Performance Improvement for Query with Constant Boolean Predicates</u>	54
<u>3.1.18 Various Problems When LOCK PARTITIONING IS ENABLED</u>	54
<u>3.1.19 Wrong Result from Query with Zigzag Strategy</u>	55
<u>3.1.20 Query Slows Down Significantly After Upgrade from Rdb Release 7.0.7.3</u>	58
<b><u>3.2 SQL Errors Fixed</u></b>	<b>65</b>
<u>3.2.1 Unexpected Column Ordering Generated by ALTER TABLE ... ALTER COLUMN Statements</u>	65

# Table of Contents

<b><u>3.2 SQL Errors Fixed</u></b>	
<u>3.2.2 Unexpected Errors when Accessing Declared Local Temporary Tables</u>	66
<u>3.2.3 SQL\$MOD Bugcheck for Undefined Function Parameter</u>	66
<u>3.2.4 DDL Statements Caused Bugchecks with Various Exceptions</u>	67
<b><u>3.3 RMU Errors Fixed</u></b>	<b>69</b>
<u>3.3.1 RMU/COPY Would Increment Roll-Forward Sequence Number</u>	69
<u>3.3.2 RMU /UNLOAD Output File Maximum Record Size</u>	71
<u>3.3.3 RMU Extract Did Not Generate Correct Routine References for Multischema Databases</u>	71
<b><u>3.4 Row Cache Errors Fixed</u></b>	<b>73</b>
<u>3.4.1 Hangs With Row Cache Hash Latches Using Rdb Release 7.1.4.1.1</u>	73
<b><u>3.5 Hot Standby Errors Fixed</u></b>	<b>74</b>
<u>3.5.1 RDM\$BIND HOT NETWORK RETRY COUNT Not Consistently Used</u>	74
<u>3.5.2 AIJSIGNATURE Error Not Always Returned When Journals Different</u>	74
<u>3.5.3 Standby Journal Disk No Longer Has to Have Same Cluster Size as Master</u>	76
<u>3.5.4 Cannot Start Hot Standby if Incremental Restore Done</u>	77
<u>3.5.5 RMU/REPLICATE AFTER Qualifiers Ignored</u>	77
<b><u>Chapter 4 Software Errors Fixed in Oracle Rdb Release 7.1.4.1</u></b>	<b>78</b>
<b><u>4.1 Software Errors Fixed That Apply to All Interfaces</u></b>	<b>79</b>
<u>4.1.1 Memory Leak With RMU /OPEN /STATISTICS=IMPORT</u>	79
<u>4.1.2 Query With Two ORDER BY Clauses Returns Wrong Result</u>	79
<u>4.1.3 Database Shutdown Interrupted by Re-attach May Cause Lost ALS Process</u>	81
<u>4.1.4 Query With BETWEEN Clause Slows Down Using Index Full Scan</u>	81
<u>4.1.5 Database Recovery Process May Fail When AIJs are Full</u>	83
<u>4.1.6 Count Distinct(fld) May Fail When Sorted Ranked Indexes are Used</u>	84
<u>4.1.7 Left Outer Join View Query With Constant Columns Returns Wrong Result</u>	84
<u>4.1.8 Bugchecks at DIOMARK\$NEW SNAP PAGE + 000000D0 When Area Added Online</u>	85
<u>4.1.9 Query with GROUP BY and ORDER BY Returned Rows in the Wrong Order</u>	86
<u>4.1.10 Query with GROUP BY, ORDER BY Returned Rows in the Wrong Order</u>	88
<u>4.1.11 Truncating Empty Table Leaves Uncommitted Transaction in Journal</u>	91
<u>4.1.12 Bugchecks in AIJUTL\$FREE DIRTY ARBS When Journals Full</u>	92
<u>4.1.13 Query With Shared Expressions in OR Predicates Returns Wrong Result</u>	93
<u>4.1.14 Failed Users Not Recovered if DBR Startup Fails</u>	95
<u>4.1.15 Various Errors or Corruption of Ranked Indexes</u>	95
<u>4.1.16 Wrong Results Generated by Query With Common Boolean Elements</u>	96
<u>4.1.17 Wrong Result From Query With Common Join Booleans in OR</u>	97
<u>4.1.18 Wrong Result Selecting From a Derived Table of UNION Clause</u>	98
<u>4.1.19 Incorrect Foreign Key Constraint Behavior on Update</u>	100
<u>4.1.20 Bugchecks Accessing a REAL or DOUBLE PRECISION Column</u>	102
<u>4.1.21 Bugchecks in PSII2SPLITNODE When Using Ranked Indexes</u>	103
<u>4.1.22 Wrong Result from UNION Query with Outer Join Leg</u>	103
<u>4.1.23 COSI MEM FREE VMLIST Bugcheck with Vertical Partitioning</u>	110
<u>4.1.24 Bugcheck from INSERT With Partition Index</u>	111
<u>4.1.25 Journals Not Initialized After Backup if Backing Up to Tape Device</u>	111

# Table of Contents

<b>4.1 Software Errors Fixed That Apply to All Interfaces</b>	
4.1.26 Constant Snapshot File Growth.....	112
4.1.27 Select Count Query with Host Variable Returns Wrong Result.....	113
4.1.28 UNION Join Query with Host Variable in the Predicate Returns Wrong Result.....	114
4.1.29 Bugcheck in COSI MEM FREE VMLIST After Update of Ranked Index.....	116
4.1.30 ILLPAGCNT Exception Reading Large Table with a Dynamic Tactic.....	117
<b>4.2 SQL Errors Fixed.....</b>	<b>118</b>
4.2.1 SQL Precompiler (Pascal) Generates Incorrect Definition for SQL TINYINT Type.....	118
4.2.2 LOCK TABLE May Bugcheck if DROP TABLE Appears in Same Transaction.....	118
4.2.3 CREATE VIEW May Fail With a "Deadlock on Client" Error.....	119
4.2.4 TRUNCATE TABLE Did Not Release Strong Lock on Table Until DISCONNECT.....	119
4.2.5 COMMENT ON COLUMN Failed When Applied to a View Definition.....	119
4.2.6 Unexpected SQL-F-CURALROPE Following Compound Statement or CALL Statement.....	120
4.2.7 Unexpected Results When Using Host Variables in Subselects.....	120
4.2.8 Unexpected Bugcheck Reported When LIST OF BYTE VARYING Column has NOT NULL Constraint.....	121
4.2.9 DEFAULT Expression Fails for Declared Temporary Tables.....	122
4.2.10 DEFAULT Inherited from Domain now Displayed by SHOW TABLE.....	123
4.2.11 Dynamic SQL Rounds Results from Division Operator.....	123
4.2.12 SQL Incorrectly Truncated Multi-octet Characters when Using GB18030 and UTF8 Character Sets.....	124
4.2.13 New COMMIT EVERY Clause Added to IMPORT DATABASE Command.....	124
4.2.14 Unexpected SQL-F-BADBLOB Reported by SQL IMPORT.....	126
4.2.15 Connection Name Longer than 31 Octets Mishandled.....	127
4.2.16 Loss of NULL Setting for Imported LIST OF BYTE VARYING Columns.....	127
<b>4.3 RDO and RDML Errors Fixed.....</b>	<b>129</b>
4.3.1 Unexpected NOT LARDY Following LOCK CONFLICT Exception.....	129
<b>4.4 RMU Errors Fixed.....</b>	<b>130</b>
4.4.1 RMU Extract Might Extract Incomplete Routine Definition.....	130
4.4.2 Unexpected ACCVIO and BUGCHECK from RMU Extract.....	130
4.4.3 RMU Load Did Not Handle UNSPECIFIED Character Set in RRD File.....	131
4.4.4 RMU/CONVERT Ignored Storage Maps Defined for Optional System Tables.....	131
4.4.5 RMU /UNLOAD From Remote Database Specification.....	132
4.4.6 RMU/SHOW AFTER/BACKUP May Stall When AIJs Are Full.....	132
4.4.7 Problem with LITERAL Character Set and Embedded Quotes in RMU EXTRACT.....	133
<b>4.5 LogMiner Errors Fixed.....</b>	<b>134</b>
4.5.1 Incorrect Elimination of AIJ When Using RMU /UNLOAD /AFTER JOURNAL /ORDER AIJ FILES /RESTART.....	134
4.5.2 RMU /UNLOAD /AFTER JOURNAL /[NO]SYMBOLS.....	134
4.5.3 RMU /UNLOAD /AFTER JOURNAL Incorrect Settings in Null Bit Vector.....	134
4.5.4 RMU /UNLOAD /AFTER JOURNAL Field Order Clarification.....	135

# Table of Contents

<b><u>4.6 Row Cache Errors Fixed</u></b> .....	<b>137</b>
4.6.1 <u>RMU /CLOSE /ABORT=DELPRC /WAIT Hang</u> .....	137
4.6.2 <u>Long Running Transaction Hangs After RMU/CHECKPOINT</u> .....	137
<b><u>4.7 Hot Standby Errors Fixed</u></b> .....	<b>139</b>
4.7.1 <u>Excessive CPU Consumed by LRS Process</u> .....	139
4.7.2 <u>LRS Bugchecks in DIOLAREA\$SCAN ABM CHAIN</u> .....	139
<b><u>Chapter 5 Enhancements Provided in Oracle Rdb Release 7.1.4.2</u></b> .....	<b>142</b>
<b><u>5.1 Enhancements Provided in Oracle Rdb Release 7.1.4.2</u></b> .....	<b>143</b>
5.1.1 <u>/TRANSPORT Added to RMU/REPLICATE AFTER</u> .....	143
5.1.2 <u>CASCADE Option Now Supported by DROP INDEX</u> .....	143
5.1.3 <u>Oracle Rdb Release 7.1.x.x New Features Document Added</u> .....	144
5.1.4 <u>READER THREAD RATIO and RMU BACKUP</u> .....	144
5.1.5 <u>RMU/DUMP/AFTER JOURNAL AREA, PAGE and LINE QUALIFIERS</u> .....	145
<b><u>Chapter 6 Enhancements Provided in Oracle Rdb Release 7.1.4.1</u></b> .....	<b>148</b>
<b><u>6.1 Enhancements Provided in Oracle Rdb Release 7.1.4.1</u></b> .....	<b>149</b>
6.1.1 <u>Support Added for ANSI C Comments</u> .....	149
6.1.2 <u>New Rdb Character Set GB18030</u> .....	149
6.1.3 <u>New GET DIAGNOSTICS Keyword</u> .....	149
6.1.4 <u>Buffer Memory Now Exported/Imported</u> .....	150
6.1.5 <u>RMU /UNLOAD Qualifier /REOPEN COUNT</u> .....	150
6.1.6 <u>New DEFAULTS Qualifier Added to RMU Extract</u> .....	150
6.1.7 <u>New RESTART WITH Clause for ALTER SEQUENCE</u> .....	152
6.1.8 <u>ALTER VIEW Statement</u> .....	152
<b><u>Chapter 7 Documentation Corrections, Additions and Changes</u></b> .....	<b>157</b>
<b><u>7.1 Documentation Corrections</u></b> .....	<b>158</b>
7.1.1 <u>Incorrect Example Under RMU Unload Command</u> .....	158
7.1.2 <u>RDMSBIND MAX DBR COUNT Documentation Clarification</u> .....	158
7.1.3 <u>Database Server Process Priority Clarification</u> .....	159
7.1.4 <u>Explanation of SQL\$INT in a SQL Multiversion Environment and How to Redefine SQL\$INT</u> .....	160
7.1.5 <u>Documentation Omitted Several Reserved Words</u> .....	161
7.1.6 <u>Using Databases from Releases Earlier Than V6.0</u> .....	161
7.1.7 <u>New RMU/BACKUP Storage Area Assignment With Thread Pools</u> .....	162
7.1.8 <u>RDMSBIND LOCK TIMEOUT INTERVAL Overrides the Database Parameter</u> .....	162
7.1.9 <u>New Request Options for RDO, RDBPRE and RDB\$INTERPRET</u> .....	162
<b><u>7.2 Address and Phone Number Correction for Documentation</u></b> .....	<b>166</b>
<b><u>7.3 Online Document Format and Ordering Information</u></b> .....	<b>167</b>

# Table of Contents

<b><u>7.4 New and Changed Features in Oracle Rdb Release 7.1.....</u></b>	<b>168</b>
<u>7.4.1 PERSONA is Supported in Oracle SQL/Services.....</u>	168
<u>7.4.2 NEXTVAL and CURRVAL Pseudocolumns Can Be Delimited Identifiers.....</u>	168
<u>7.4.3 Only=select list Qualifier for the RMU Dump After Journal Command.....</u>	168
<b><u>7.5 Oracle Rdb7 and Oracle CODASYL DBMS Guide to Hot Standby Databases.....</u></b>	<b>170</b>
<u>7.5.1 Restrictions Lifted on After-Image Journal Files.....</u>	170
<u>7.5.2 Changes to RMU Replicate After Journal ... Buffer Command.....</u>	170
<u>7.5.3 Unnecessary Command in the Hot Standby Documentation.....</u>	171
<u>7.5.4 Change in the Way RDMAIJ Server is Set Up in UCX.....</u>	171
<u>7.5.5 CREATE INDEX Operation Supported for Hot Standby.....</u>	172
<b><u>7.6 Oracle Rdb7 for OpenVMS Installation and Configuration Guide.....</u></b>	<b>173</b>
<u>7.6.1 Suggestion to Increase GH RSRVPGCNT Removed.....</u>	173
<u>7.6.2 Prerequisite Software.....</u>	173
<u>7.6.3 Defining the RDBSERVER Logical Name.....</u>	173
<b><u>7.7 Guide to Database Design and Definition.....</u></b>	<b>175</b>
<u>7.7.1 Lock Timeout Interval Logical Incorrect.....</u>	175
<u>7.7.2 Example 4-13 and Example 4-14 Are Incorrect.....</u>	175
<b><u>7.8 Oracle RMU Reference Manual, Release 7.0.....</u></b>	<b>176</b>
<u>7.8.1 RMU Unload After Journal Null Bit Vector Clarification.....</u>	176
<u>7.8.2 New Transaction Mode Qualifier for Oracle RMU Commands.....</u>	178
<u>7.8.3 RMU Server After Journal Stop Command.....</u>	180
<u>7.8.4 Incomplete Description of Protection Qualifier for RMU Backup After Journal Command... 180</u>	180
<u>7.8.5 RMU Extract Command Options Qualifier.....</u>	180
<u>7.8.6 RDMSSNAP QUIET POINT Logical is Incorrect.....</u>	180
<u>7.8.7 Using Delta Time with RMU Show Statistics Command.....</u>	180
<b><u>7.9 Oracle Rdb7 Guide to Database Performance and Tuning.....</u></b>	<b>182</b>
<u>7.9.1 Dynamic OR Optimization Formats.....</u>	182
<u>7.9.2 Oracle Rdb Logical Names.....</u>	182
<u>7.9.3 Waiting for Client Lock Message.....</u>	182
<u>7.9.4 RDMS\$TTB HASH SIZE Logical Name.....</u>	184
<u>7.9.5 Error in Updating and Retrieving a Row by Dbkey Example 3-22.....</u>	184
<u>7.9.6 Error in Calculation of Sorted Index in Example 3-46.....</u>	185
<u>7.9.7 Documentation Error in Section C.7.....</u>	186
<u>7.9.8 Missing Tables Descriptions for the RDBEXPERT Collection Class.....</u>	186
<u>7.9.9 Missing Columns Descriptions for Tables in the Formatted Database.....</u>	187
<u>7.9.10 A Way to Find the Transaction Type of a Particular Transaction Within the Trace Database.....</u>	194
<u>7.9.11 Using Oracle TRACE Collected Data.....</u>	194
<u>7.9.12 AIP Length Problems in Indexes that Allow Duplicates.....</u>	196
<u>7.9.13 RDM\$BIND MAX DBR COUNT Documentation Clarification.....</u>	197

# Table of Contents

<b>7.10 Oracle Rdb7 Guide to SQL Programming</b> .....	<b>199</b>
7.10.1 Location of Host Source File Generated by the SQL Precompiler.....	199
7.10.2 Remote User Authentication.....	200
7.10.3 Additional Information About Detached Processes.....	200
<b>7.11 Guide to Using Oracle SQL/Services Client APIs</b> .....	<b>202</b>
<b>Chapter 8 Known Problems and Restrictions</b> .....	<b>203</b>
<b>8.1 Known Problems and Restrictions in All Interfaces</b> .....	<b>204</b>
8.1.1 Domain-qualified TCP/IP Node Names in Distributed Transactions.....	204
8.1.2 Some SQL Dialect-required Warnings not Delivered.....	205
8.1.3 Partitioned Index with Descending Column and Collating Sequence.....	206
8.1.4 RDO IMPORT Does Not Support FORWARD REFERENCES Created by SQL EXPORT.....	207
8.1.5 New Attributes Saved by RMU/LOAD Incompatible With Prior Versions.....	208
8.1.6 RDMS-E-RTNSBC INITERR. Cannot init. external routine server site executor.....	209
8.1.7 SYSTEM-F-INSFMEM Fatal Error With SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED in Galaxy Environment.....	209
8.1.8 Oracle Rdb and OpenVMS ODS-5 Volumes.....	210
8.1.9 Optimization of Check Constraints.....	210
8.1.10 Using Databases from Releases Earlier Than V6.0.....	213
8.1.11 Carryover Locks and NOWAIT Transaction Clarification.....	213
8.1.12 Unexpected Results Occur During Read-Only Transactions on a Hot Standby Database.....	213
8.1.13 Both Application and Oracle Rdb Using SYSS\$HIBER.....	214
8.1.14 Bugcheck Dump Files with Exceptions at COSI CHF SIGNAL.....	215
8.1.15 Read-only Transactions Fetch AIP Pages Too Often.....	215
8.1.16 Row Cache Not Allowed While Hot Standby Replication is Active.....	216
8.1.17 Excessive Process Page Faults and other Performance Considerations During Oracle Rdb Sorts.....	216
8.1.18 Control of Sort Work Memory Allocation.....	217
8.1.19 The Halloween Problem.....	218
<b>8.2 SQL Known Problems and Restrictions</b> .....	<b>220</b>
8.2.1 Interchange File (RBR) Created by Oracle Rdb Release 7.1 Not Compatible With Previous Releases.....	220
8.2.2 System Relation Change for International Database Users.....	220
8.2.3 Single Statement LOCK TABLE is Not Supported for SQL Module Language and SQL Precompiler.....	220
8.2.4 Multistatement or Stored Procedures May Cause Hangs.....	221
8.2.5 Use of Oracle Rdb from Shareable Images.....	222
<b>8.3 Oracle RMU Known Problems and Restrictions</b> .....	<b>223</b>
8.3.1 RMU/BACKUP MAX FILE SIZE Option Has Been Disabled.....	223
8.3.2 RMU Convert Fails When Maximum Relation ID is Exceeded.....	223
8.3.3 RMU Unload /After Journal Requires Accurate AIP Logical Area Information.....	224
8.3.4 Do Not Use HYPERSORT with RMU Optimize After Journal Command.....	225
8.3.5 Changes in EXCLUDE and INCLUDE Qualifiers for RMU Backup.....	225

# Table of Contents

<b><u>8.3 Oracle RMU Known Problems and Restrictions</u></b>	
<u>8.3.6 RMU Backup Operations Should Use Only One Type of Tape Drive</u>	226
<u>8.3.7 RMU/VERIFY Reports PGSPAMENT or PGSPMCLST Errors</u>	226
<b><u>8.4 Known Problems and Restrictions in All Interfaces for Release 7.0 and Earlier</u></b>	<b>228</b>
<u>8.4.1 Converting Single-File Databases</u>	228
<u>8.4.2 Row Caches and Exclusive Access</u>	228
<u>8.4.3 Exclusive Access Transactions May Deadlock with RCS Process</u>	228
<u>8.4.4 Strict Partitioning May Scan Extra Partitions</u>	228
<u>8.4.5 Restriction When Adding Storage Areas with Users Attached to Database</u>	229
<u>8.4.6 Support for Single-File Databases to Be Dropped in a Future Release</u>	229
<u>8.4.7 Multiblock Page Writes May Require Restore Operation</u>	230
<u>8.4.8 Replication Option Copy Processes Do Not Process Database Pages Ahead of an Application</u>	230
<b><u>8.5 SQL Known Problems and Restrictions for Oracle Rdb Release 7.0 and Earlier</u></b>	<b>232</b>
<u>8.5.1 SQL Does Not Display Storage Map Definition After Cascading Delete of Storage Area</u>	232
<u>8.5.2 ARITH EXCEPT or Incorrect Results Using LIKE IGNORE CASE</u>	232
<u>8.5.3 Different Methods of Limiting Returned Rows from Queries</u>	233
<u>8.5.4 Suggestions for Optimal Use of SHARED DATA DEFINITION Clause for Parallel Index Creation</u>	234
<u>8.5.5 Side Effect When Calling Stored Routines</u>	235
<u>8.5.6 Considerations When Using Holdable Cursors</u>	236
<u>8.5.7 AIJSERVER Privileges</u>	237

# Oracle® Rdb for OpenVMS

# Release Notes

Release 7.1.4.3

---

# December 2005

Oracle Rdb Release Notes, Release 7.1.4.3 for OpenVMS

Copyright © 1984, 2005 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, except to the extent required to obtain interoperability with other independently created software or as specified by law, 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 Hot Standby, LogMiner for Rdb, Oracle CDD/Repository, Oracle CODASYL DBMS, Oracle Expert, Oracle Rdb, Oracle RMU, Oracle RMUwin, Oracle SQL/Services, Oracle Trace, and Rdb7 are trademark or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

---

# Contents

---

# Preface

# Purpose of This Manual

This manual contains release notes for Oracle Rdb Release 7.1.4.3. The notes describe changed and enhanced features; upgrade and compatibility information; new and existing software problems and restrictions; and software and documentation corrections.

# Intended Audience

This manual is intended for use by all Oracle Rdb users. Read this manual before you install, upgrade, or use Oracle Rdb Release 7.1.4.3.

# Document Structure

This manual consists of the following chapters:

<u>Chapter 1</u>	Describes how to install Oracle Rdb Release 7.1.4.3.
<u>Chapter 2</u>	Describes software errors corrected in Oracle Rdb Release 7.1.4.3.
<u>Chapter 3</u>	Describes software errors corrected in Oracle Rdb Release 7.1.4.2.
<u>Chapter 4</u>	Describes software errors corrected in Oracle Rdb Release 7.1.4.1.
<u>Chapter 5</u>	Describes enhancements introduced in Oracle Rdb Release 7.1.4.2.
<u>Chapter 6</u>	Describes enhancements introduced in Oracle Rdb Release 7.1.4.1.
<u>Chapter 7</u>	Provides information not currently available in the Oracle Rdb documentation set.
<u>Chapter 8</u>	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb Release 7.1.4.3.

---

# Chapter 1

## Installing Oracle Rdb Release 7.1.4.3

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

---

### NOTE

*All Oracle Rdb Release 7.1 kits are full kits. There is no requirement to install any prior release of Oracle Rdb when installing new Rdb Release 7.1 kits.*

---

# 1.1 Alpha EV7 Processor Support

For this release of Oracle Rdb, the Alpha EV7 (also known as the Alpha 21364) processor is the newest processor supported.

## 1.2 Oracle Rdb V7.1 Version Numbering Enhancement

Previously, the Oracle Rdb version number was specified as 4 digits (for example, version "7.1.0.2"). Starting with Oracle Rdb Release 7.1.1, an additional, fifth, digit has been added to the kit version number. This new digit is intended to indicate an optimization level of the Rdb software. The use of this new digit is to indicate a "generic" kit (final digit of zero) for all Alpha processors or a "performance" kit that will run on a subset of the supported platforms (final digit of 1). In the future, additional values may be specified to indicate other performance or platform options.

For Oracle Rdb Release 7.1.4.3, the two kits are 7.1.4.3.0 (compiled for all Alpha processor types) and 7.1.4.3.1 (compiled for EV56 and later Alpha processors). These kits offer identical functionality and differ only in a potential performance difference.

## 1.3 Requirements

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

- Oracle Rdb must be shutdown before you install this update kit. That is, the command file `SYS$STARTUP:RMONSTOP71.COM` should be executed before proceeding with this installation. If you have an OpenVMS cluster, you must shutdown the Rdb 7.1 monitor on all nodes in the cluster before proceeding.
- The installation requires approximately 280,000 blocks for OpenVMS Alpha systems.
- If you are running Hot Standby and you are upgrading from a version of Oracle Rdb 7.1 prior to 7.1.1, you must install this kit on both the master and the standby systems prior to restarting Hot Standby. This requirement is necessary due to changes to the message format used to transmit journal state information from the master to the standby system.

## 1.4 Invoking VMSINSTAL

To start the installation procedure, invoke the VMSINSTAL command procedure as in the following examples.

To install the Oracle Rdb for OpenVMS Alpha kit that is compiled to run on all Alpha platforms:

```
@SYS$UPDATE:VMSINSTAL RDBV71430AM device-name OPTIONS N
```

To install the Oracle Rdb for OpenVMS Alpha kit that is performance targeted for Alpha EV56 and later platforms:

```
@SYS$UPDATE:VMSINSTAL RDBV71431AM device-name OPTIONS N
```

### *device-name*

Use the name of the device on which the media is mounted. If the device is a disk drive, you also need to specify a directory. For example: *DKA400:[RDB.KIT]*

### *OPTIONS N*

This parameter prints the release notes.

The full Oracle Rdb Release 7.1 Installation Guide is also available on MetaLink in Adobe Acrobat PDF format:

```
Top Tech Docs\Oracle Rdb\Documentation\Rdb 7.1 Installation and Configuration  
Guide
```

## 1.5 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 Rdb installed will probably not be usable.

## 1.6 After Installing Oracle Rdb

This update provides a new Oracle Rdb Oracle TRACE facility definition. Any Oracle TRACE selections that reference Oracle Rdb will need to be redefined to reflect the new facility version number for the updated Oracle Rdb facility definition, "RDBVMSV7.1".

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

The installation procedure inserts the Oracle Rdb facility definition into a library file called EPC\$FACILITY.TLB. To be able to collect Oracle Rdb 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=RDBVMSV7.1 -  
_ $ /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 the process executing the INSERT DEFINITION command must use the version of Oracle Rdb that matches the version used to create the Oracle TRACE administration database or the INSERT DEFINITION command will fail.

## 1.7 Spurious SYSVERDIF Message During Installation

When installing Oracle Rdb on an OpenVMS V8.2 Alpha system, depending on the previous version of Oracle Rdb installed and how Oracle Rdb was shut down prior to the installation, a message similar to the following may be displayed during the installation procedure:

```
%INSTALL-E-FAIL, failed to REPLACE entry for  
DISK$VMS82:<SYS0.SYSCOMMON.SYSLIB>RDMXSMP71.EXE  
-INSTALL-E-SYSVERDIF, system version mismatch - please relink
```

This message may be safely ignored. Using the RMONSTOP.COM (for standard installations) or RMONSTOP71.COM (for multi-version installations) procedure to shut down Oracle Rdb prior to the installation may help avoid the message.

## 1.8 Patches for OpenVMS V7.3–1

Several problems that affect installations using Oracle Rdb on OpenVMS V7.3–1 are corrected in patch kits available from HP OpenVMS support. Oracle recommends that you consult with Hewlett–Packard and install these patch kits (or their replacements) to correct or avoid the following problems:

- VMS731\_SYS–V0400 corrects the following problems seen with Oracle Rdb:
  - ◆ When using Oracle Rdb Galaxy support, or memory–resident global sections, processes enter a permanent RWAST state at image exit. The system must be rebooted to remove the process and continue normal operations. Note that when using Oracle Rdb Release 7.1.2 databases with SHARED MEMORY IS PROCESS RESIDENT attribute, the Row Cache feature and caches with the SHARED MEMORY IS SYSTEM, LARGE MEMORY IS ENABLED, or RESIDENT attributes, or in an OpenVMS Galaxy configuration with Oracle Rdb Galaxy support enabled, you are at an elevated risk of experiencing this problem. Configurations that do not have this patch, or it's future replacement, applied will not be supported by Oracle if the SHARED MEMORY IS PROCESS RESIDENT, the Row Cache, or Galaxy support features are in use. If you are not using these features, then the patch or it's replacement is not mandatory. However, Oracle still strongly recommends that it be used.
  - ◆ Applications using the Oracle Rdb Row Cache or AIJ Log Server (ALS) features would sometimes have their server processes hang in HIB (hibernate) state.
- VMS731\_SYSLOA–V0100 corrects the following problem seen with Oracle Rdb:
  - ◆ In an OpenVMS cluster environment, unreported deadlocks and hangs can occur. This problem is sometimes characterized by an Oracle Rdb blocking lock incorrectly being shown as owned by the system (in other words, with a zero PID).

# 1.9 Oracle Rdb Release 7.1.4.3.1 Optimized for Alpha EV56 (21164A Processor Chip) and Later Platforms

Oracle will be releasing Oracle Rdb 7.1 and later kits in parallel build streams – a "generic" kit that will run on all certified and supported Alpha platforms as well as a "performance" kit that will run on a subset of the supported platforms. The performance kit is intended for those customers with "newer" Alpha processor chips who need higher levels of performance than are offered by the generic kits. The performance kits are otherwise functionally identical to the generic kits.

Oracle will continue to release both types of kits for Oracle Rdb Release 7.1 as long as there is significant customer interest in the generic kit.

For improved performance on current generation Alpha processors, Oracle Rdb Release 7.1.4.3.1 is compiled explicitly for Alpha EV56 and later systems. This version of Oracle Rdb requires a system with a minimum Alpha processor chip of EV56 and a maximum processor chip of Alpha EV7 (known as the Alpha 21364).

Oracle Rdb Release 7.1.4.3.1 is functionally equivalent to Oracle Rdb Release 7.1.4.3.0 and was built from the same source code. The only difference is a potentially improved level of performance. Oracle Rdb Releases 7.1.4.3.0 and 7.1.4.3.1 are certified on all supported Alpha processor types (up to and including the Alpha EV7 processor).

In Release 7.1.4.3.1, Oracle Rdb is explicitly compiled for EV56 and later Alpha processors such that the generated instruction stream can utilize the byte/word extension (BWX) of the Alpha architecture. Additionally, this kit is compiled with instruction tuning biased for performance of Alpha EV6 and later systems that support quad-issue instruction scheduling.

Note that you should not install Release 7.1.4.3.1 of Oracle Rdb on Alpha EV4, EV45 or EV5 systems. These processor types do not support the required byte/word extension (BWX) of the Alpha architecture. Also ensure that all systems in a cluster sharing the system disk are using a minimum of the Alpha EV56 processor.

To easily determine the processor type of a running OpenVMS Alpha system, use the CLUE CONFIG command of the OpenVMS System Dump Analyzer utility (accessed with the ANALYZE/SYSTEM command). The "CPU TYPE" field indicates the processor type as demonstrated in the following example from an HP AlphaServer GS140 6/525 system with an EV6 (21264) processor:

```
$ ANALYZE/SYSTEM
SDA> CLUE CONFIG
System Configuration:
.
.
.
Per-CPU Slot Processor Information:
CPU ID      00                CPU State   rc,pa,pp,cv,pv,pmv,pl
CPU Type    EV6   Pass 2.3 (21264)
PAL Code    1.96-1                Halt PC     00000000.20000000
.
.
.
```

## 1.9.1 AlphaServer 4000 EV56 299Mhz Not Supported by Oracle Rdb Release Optimized for Alpha EV56 Processor

Oracle Rdb releases that are optimized for the Alpha EV56 and later processors are not able to run on the AlphaServer 4000 with the 299Mhz EV56 processor. Though this CPU claims to be an EV56, it does not, in fact, implement the byte/word instruction set as required.

According to information on the HP web site, this problem may be present in the AlphaServer 4000 or 4100 systems with a processor module of KN304-FA or KN304-FB. The systems effected appear to include the AlphaServer 4x00 5/300 pedestal, cabinet and rackmount systems: DA-51FAB-ED/-FD/-GB or DA-53GEB-CA/-EA/-FA/-GA.

The indicated CPU is not able to run Oracle Rdb releases that are optimized for the Alpha EV56 and later processors. This effects Oracle Rdb Releases optimized for the Alpha EV56 and later processors.

Possible workarounds include updating the system to an EV56 module for the AlphaServer 4x00 that is later than the KN304-FA or FB (ie a clock speed greater than 300Mhz). Some of the possible modules would include: KN304-AA 400mhz, KN304-DA 466mhz, B3005-CA 533mhz, B3006-EB 600mhz.

Otherwise, an Oracle Rdb release that is not optimized for the Alpha EV56 and later processors must be used (such as Oracle Rdb Releases 7.1.4.3.0)

Please contact your HP AlphaServer hardware vendor for additional information.

## 1.10 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 8.2-x is the maximum supported version of OpenVMS.

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.

## 1.11 VMS\$MEM\_RESIDENT\_USER Rights Identifier Required

Oracle Rdb Version 7.1 introduced additional privilege enforcement for the database or row cache attributes RESIDENT, SHARED MEMORY IS SYSTEM and LARGE MEMORY IS ENABLED. If a database utilizes any of these features, then the user account that opens the database must be granted the VMS\$MEM\_RESIDENT\_USER rights identifier.

Oracle recommends that the RMU/OPEN command be used when utilizing these features.

---

# **Chapter 2**

## **Software Errors Fixed in Oracle Rdb Release 7.1.4.3**

This chapter describes software errors that are fixed by Oracle Rdb Release 7.1.4.3.

## 2.1 Software Errors Fixed That Apply to All Interfaces

### 2.1.1 Various Bugchecks Due to Memory Corruption

Bugs 4762619 and 4742021

If an application attached to multiple databases, and those databases did not have an identical number of TSNBLKs allocated to each database, then bugchecks related to memory corruption could occur. Exceptions like the following were typically encountered:

```
***** Exception at 0105C084 : COSI_MEM_GET_VM + 000000E4
***** Exception at 01C96394 : COSI_MEM_GET_VM_2 + 000000A4
***** Exception at 014F7764 : COSI_MEM_FREE_VMLIST + 00000094
```

A TSNBLK is an internal data structure allocated in the database root (.RDB) file. The number of TSNBLKs allocated is dependent on the number of users and the number of cluster nodes declared for the database. Thus a database that is configured for one cluster node will not have the same number of TSNBLKs allocated as a database configured for 16 cluster nodes.

This problem was introduced in Oracle Rdb Releases 7.0.8 and 7.1.4.1.

The problem can be avoided by configuring the databases accessed by the application such that they have the same limits for the number of users and cluster nodes.

This problem has been corrected in Oracle Rdb Release 7.1.4.3.

### 2.1.2 Reserved Sequence Slots Not Properly Initialized

Bug 4724663

When altering a database to reserve additional sequence "slots", it was possible that the newly reserved sequence slots in the root file were not correctly initialized. This could lead to errors being reported by RMU/VERIFY.

The following example demonstrates this problem.

```
$!
$ SQL$
    ALTER DATABASE FILENAME MF_PERSONNEL
        RESERVE 70 STORAGE AREAS
        RESERVE 70 SEQUENCES;
%RDMS-W-DOFULLBCK, full database backup should
    be done to ensure future recovery
%RDMS-W-DOFULLBCK, full database backup should
    be done to ensure future recovery
    EXIT;
$!
$ RMU /VERIFY /ALL /NOLOG MF_PERSONNEL
%RMU-E-NOSEQROW, sequence id 33 has an entry in
```

```

the root file but no row in RDB$SEQUENCES
%RMU-E-NOSEQROW, sequence id 35 has an entry in
the root file but no row in RDB$SEQUENCES
%RMU-E-NOSEQROW, sequence id 45 has an entry in
the root file but no row in RDB$SEQUENCES
%RMU-E-NOSEQROW, sequence id 65 has an entry in
the root file but no row in RDB$SEQUENCES
%RMU-E-NOSEQROW, sequence id 67 has an entry in
the root file but no row in RDB$SEQUENCES
%RMU-E-NOSEQROW, sequence id 77 has an entry in
the root file but no row in RDB$SEQUENCES
%RMU-E-NOSEQROW, sequence id 97 has an entry in
the root file but no row in RDB$SEQUENCES
%RMU-E-NOSEQROW, sequence id 99 has an entry in
the root file but no row in RDB$SEQUENCES
%RMU-E-NOSEQROW, sequence id 109 has an entry in
the root file but no row in RDB$SEQUENCES
$!

```

This problem has been corrected in Oracle Rdb Release 7.1.4.3. The reserved sequence blocks in the root file are now initialized correctly.

## 2.1.3 User Attach Stalls Waiting for ALS Startup

Bug 4767826

It was possible for a user attach request to stall indefinitely when the following occurred:

1. A user had updated the database from one node.
2. An RMU/SHOW STATISTICS session had been started on another node. The database was not manually opened and no other users were attached to the database on that node.
3. The database was forced closed on the first node due to a node failure or by an RMU/CLOSE/ABORT=DELPRC.
4. Before the failure of the first node could be recovered by the second node, a user attach request was made.

When the above sequence of events occurred, the monitor would show the attach request as pending and would never complete the attach request. For example:

```

$ RMU/SHOW SYSTEM
Oracle Rdb X7.1-00 on node CLYPPR 29-NOV-2005 13:18:13.07
  - monitor started 29-NOV-2005 08:25:19.67 (uptime 0 04:52:53)
  - monitor log filename is "DEV:[DIR]RDMMON.LOG"

database DEV:[DIR]MF_PERSONNEL.RDB;1
  - opened 29-NOV-2005 11:57:31.54 (elapsed 0 01:20:41)
  - current after-image journal file is
    DEV:[DIR]AIJ_4.AIJ;1
  - 1 pending database user on this node

```

This problem can be demonstrated using the following commands. Four different sessions are needed; two on each node.

Session 1, Node 1:

```
$ SQL$
```

## Oracle® Rdb for OpenVMS

```
SQL> ATTACH 'FILENAME MF_PERSONNEL';  
SQL> DELETE FROM EMPLOYEES;  
100 rows deleted  
SQL>
```

Session 1, Node 2:

```
$ RMU/SHOW STATISTICS MF_PERSONNEL
```

Session 2, Node 1:

```
$ RMU/CLOSE/ABORT=DELPRC MF_PERSONNEL
```

Session 2, Node 2:

```
$ SQL$  
SQL> ATTACH 'FILENAME MF_PERSONNEL';
```

If the last attach request was made while the database recovery process (DBR) was active recovering the failed attach from the first node, then the attach would stall until another attach was made to the database.

This problem can be avoided by manually opening the database using the RMU/OPEN command on each node that accesses the database, or by having a normal user attach to the database prior to invoking the RMU/SHOW STATISTICS utility.

This problem has been corrected in Oracle Rdb Release 7.1.4.3.

## 2.2 SQL Errors Fixed

### 2.2.1 CREATE TABLE ... LIKE May Create Invalid NOT NULL Constraints

Bug 4724668

In prior releases of Oracle Rdb V7.1, a CREATE TABLE ... LIKE statement may not have correctly created the NOT NULL constraints that are cloned from the source table. Tables that use a source for the LIKE clause that were created with version V7.1.3 or later do not suffer from this problem.

The following example shows that a bugcheck occurs when an attempt is made to use these constraints.

```
SQL> create table TMP_TBL like EMPLOYEES;
SQL> insert into TMP_TBL default values;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DISK2:[TESTER]RDSBUGCHK.DMP;
```

A workaround to this problem is to alter the created table to drop and redefine the not null constraints. Similarly, the NOT NULL constraints on the source table could be redefined, although this will likely incur additional I/O.

```
SQL> create table TMP_TBL like EMPLOYEES;
SQL> alter table TMP_TBL
cont>      modify
cont>      (EMPLOYEE_ID NULL);
SQL> alter table TMP_TBL
cont>      modify
cont>      (EMPLOYEE_ID NOT NULL);
SQL> insert into TMP_TBL default values;
%RDB-E-INTEG_FAIL, violation of constraint TMP_TBL_EMPLOYEE_ID_NOT_NULL caused
operation to fail
-RDB-F-ON_DB, on database DISK1:[DATABASES]RTRM_CH_DB.RDB;1
SQL> rollback;
```

Alternately, you can use ALTER TABLE ... DISABLE ALL CONSTRAINTS to disable these constraints.

This problem has been corrected in Oracle Rdb Release 7.1.4.3.

### 2.2.2 STDDEV and VARIANCE Generate Wrong Values in GROUP BY Queries

Bug 4725295

In prior releases of Oracle Rdb V7.1, the VARIANCE (also VAR\_POP and VAR\_SAMP) and STDDEV (also STDDEV\_POP and STDDEV\_SAMP) statistical computations for GROUP BY queries generated the wrong results.

The following example shows the incorrect results for GROUP BY.

```
SQL> create table family
```

```

cont>      (name char(10)
cont>      ,age real);
SQL>
SQL> set display no row count;
SQL> insert into family values ('Smith',8);
SQL> insert into family values ('Smith',10);
SQL> insert into family values ('Smith',12);
SQL>
SQL> insert into family values ('Jones',1);
SQL> insert into family values ('Jones',9);
SQL> insert into family values ('Jones',20);
SQL>
SQL> insert into family values ('Brown',10);
SQL> insert into family values ('Brown',10);
SQL> insert into family values ('Brown',10);
SQL>
SQL> select name, avg (age), stddev (age)
cont>   from family
cont>   group by name;
  NAME
Brown          1.0000000E+01    0.0000000000000000E+000
Jones          1.0000000E+01    1.267543556122103E+001
Smith          1.0000000E+01    1.622754859285078E+001

```

The following output (from this release) shows the correct results.

```

SQL> select name, avg (age), stddev (age)
cont>   from family
cont>   group by name;
  NAME
Brown          1.0000000E+01    0.0000000000000000E+000
Jones          1.0000000E+01    7.788880963698615E+000
Smith          1.0000000E+01    1.632993161855452E+000
SQL>

```

These wrong results are only generated by GROUP BY queries. Therefore a workaround to the problem is to recode the query as a series of single rows (non GROUP BY).

This problem has been corrected in Oracle Rdb Release 7.1.4.3.

## 2.2.3 SQL Module Language /PROTOTYPES Now Generates INT64 for BIGINT Parameters

Bug 4721570

In prior releases of Oracle Rdb, the SQL Module Language qualifier /PROTOTYPES (or /C\_PROTOTYPES) would generate an interface definition that used pointer to long type for the SQL procedure BIGINT parameters. This caused warnings from the C compiler as shown in the following example.

```

MYPROC (sqlstate, &s, &i, &l);
.....^
%CC-W-PTRMISMATCH, In this statement, the referenced type of the pointer
value "&l" is "__int64", which is not compatible with "long"
at line number 11 in file USER2:[TESTING]SAMPLE.C;1

```

This problem has been corrected in Oracle Rdb Release 7.1.4.3. SQL Module Language now generates int64

as the type in the prototype declaration. Applications should include ints.h in their applications and also use int64 when fetching BIGINT data. This is shown in this cut down example.

```
#include <ints.h>
#include <user2:[testing]mod.h>
#include <stdlib.h>

void main ()
{
char sqlstate[5] = "00000";
short s = 0;
int i = 0;
int64 l = 0;
MYPROC (sqlstate, &s, &i, &l);
exit (0);
}
```

## 2.2.4 IMPORT DATABASE Looses SECURITY CHECKING Attributes

Bug 4751555

In prior releases of Oracle Rdb, the IMPORT DATABASE statement would discard the SECURITY CHECKING options exported from the original database. The workaround is to provide the SECURITY CHECKING clause on the IMPORT DATABASE statement.

This problem has been corrected in Oracle Rdb Release 7.1.4.3.

## 2.3 RMU Errors Fixed

### 2.3.1 RMU/BACKUP of Database With Very Large Storage Area Count

Previously, databases with a very large number of storage areas (perhaps more than 5000) might cause RMU/BACKUP or RMU/RESTORE to fail. An internal buffer could overflow the output file block size and would either write a record that could not be read during recovery or could corrupt memory and cause an ACCVIO bugcheck during the backup.

This problem has been corrected in Oracle Rdb Release 7.1.4.3. RMU/BACKUP now correctly backs up databases with up to the maximum allowed number of storage areas (presently 8190).

### 2.3.2 RMU Backup and Restore Problems With Many Storage Areas

Bug 4699757

A number of RMU backup and restore problems have been fixed which could be triggered by Oracle Rdb databases which contain large numbers of storage areas. An exact minimum number of storage areas for which these problems could occur cannot be given since variable factors are involved, such as VMS process and virtual memory resources, but an approximate minimum number of database storage areas where some of these problems could begin to occur is 1000 storage areas. Each of these problems has been fixed and is described individually below followed by an example of the problem.

- A system access violation could occur on an RMU database backup because of a virtual memory allocation problem when creating a backup summary record which needed to contain data about the very large number of storage areas that needed to be backed up.  
The following example shows this problem occurring with a database containing 6000 storage areas.

```
$ rmu/backup manyareas.rdb manyareas.rbf
%RMU-I-BUGCHKDMP, generating bugcheck dump file DEVICE:[DIRECTORY]RMUBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=04,
virtual address=FFFFFFFFA1000004, PC=00000000003CE484, PS=0000001B
%RMU-F-FATALERR, fatal error on BACKUP
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 25-OCT-2005 15:41:22.58
```

- VMS process virtual address space could be used up since a large number of restore writer threads were waiting for the system area to be restored so that each writer thread could recreate the ABM pages for the storage area it had just finished restoring. This caused new writer threads to be created for other storage areas instead of reusing existing writer threads that had completely finished restoring a storage area. Each writer thread that was created instead of reused was an additional drain on system resources. This has been fixed by making sure the system area gets backed up and restored first if there are 1000 or more database storage areas. This assures that more writer threads will be reused and not created.

The following example shows this problem occurring with a database containing 8000 storage areas.

```
$ rmu/restore/nolog/nocdd/noafter manyareas.rbf
%COSI-F-VASFULL, virtual address space full
```

## Oracle® Rdb for OpenVMS

-SYSTEM-F-ILLPAGCNT, illegal page count parameter

- For the dump of a multiple backup file set created from a database with a large number of storage areas using the /DISK\_FILE or /LIBRARIAN qualifier, the fatal error %RMU-F-BUFFERSLOST, *all buffers are lost* could occur. This was caused partly by a problem not related to a large number of storage areas. The virtual memory for buffers that were allocated for the reader thread were not allocated once but were reallocated without being freed for each new backup RBF file in the backup file set being dumped. However, this problem was much more likely to happen because of a greater use of memory resources due to the dumping of the backup of a database with a large number of storage areas.

The following example shows this problem occurring with a database containing 8000 storage areas.

```
$ rmu/backup/disk_file=writer=3/nolog manyareas.rdb -
  disk:[directory1]manyareas.rbf,-
  disk:[directory2], disk:[directory3]
$ rmu/dump/backup/disk_file disk:[directory1]manyareas.rbf,-
  disk:[directory2], disk:[directory3]
RMU-I-DMPTXT_163, No dump option selected. Performing read check.
%RMU-F-BUFFERSLOST, all buffers are lost
%RMU-F-FATALERR, fatal error on DUMP_BACKUP
%RMU-F-FTL_DUMP, Fatal error for DUMP operation at 8-NOV-2005 16:36:28.60
```

- The following error could be returned during the restore of a database with a large number of storage areas.

```
%RMU-E-INVBLKHDR, invalid block header in backup file
```

However, note that this would happen only if this was the restore of a multi-file backup RBF file set created using the /DISK\_FILE or /LIBRARIAN qualifiers. This problem was due partly to a problem unrelated to a large number of storage areas. But this problem only happened if large summary records were created at the start of each backup RBF file in the backup file set due to the need for the summary records to contain data for a large number of storage areas.

The following example shows this problem occurring with a database containing 8000 storage areas.

```
$ rmu/backup/disk_file=writer=3/nolog manyareas.rdb -
  disk:[directory1]manyareas.rbf,-
  disk:[directory2], disk:[directory3]
$ rmu/restore/disk_file=reader=3/nolog/nocdd/noafter-
  disk:[directory1]manyareas.rbf, disk:[directory2],-
  disk:[directory3]
%RMU-E-INVBLKHDR, invalid block header in backup file
```

- During a backup operation of a database with more than approximately 6000 storage areas, RMU could fail with an access violation due to an internal buffer being overrun.

These problems have been corrected in Oracle Rdb Release 7.1.4.3.

### 2.3.3 Corrections to RMU Extract

Bugs 4745056 and 4745047

This release of Oracle Rdb corrects the following problems in RMU Extract.

- RMU would incorrectly extract a subselect if it contained a UNION, EXCEPT, MINUS or INTERSECT operator.

## Oracle® Rdb for OpenVMS

- RMU would incorrectly extract an EXISTS clause when it occurred within a TRIGGER definition with a DELETE FROM action.

These problems have been corrected in Oracle Rdb Release 7.1.4.3.

## **2.4 Row Cache Errors Fixed**

### **2.4.1 RCS and User Memory Leak With Snapshots in Row Cache**

In prior releases of Oracle Rdb, when using the snapshots in row cache feature, it was possible for the RCS (Record Cache Server) or user processes to experience a memory leak when snapshot copies of rows were moved from the cache back to the database snapshot storage areas.

This problem has been corrected in Oracle Rdb Release 7.1.4.3.

## 2.5 RMU Show Statistics Errors Fixed

### 2.5.1 Excessive Erases Done for Fragmented Rows

Bug 4741940

When erasing rows that are fragmented, the "Logical Area Statistics" screen would show more rows were deleted than were actually deleted. For example, consider the following delete statement:

```
SQL> delete from table_foo where code = 'GBI';
56 rows deleted
```

In the above output, SQL showed 56 rows were deleted however the statistics screen for that table showed 105 records erased:

```
Table TABLE_FOO in RDB$SYSTEM

statistic..... rate.per.second..... total..... average.....
name..... max..... cur..... avg..... count..... per.trans....

record marked          37          0          0.2          112          112.0
record fetched        38218         0         215.1        114654        114654.0
  fragmented          45218         0         254.5        135656        135656.0
record stored          0          0          0.0           0           0.0
  fragmented          0          0          0.0           0           0.0
pages checked          0          0          0.0           0           0.0
  saved IO             0          0          0.0           0           0.0
  discarded            0          0          0.0           0           0.0
record erased          35          0          0.1          105          105.0
  fragmented          0          0          0.0           2           2.0
sequential scan        0          0          0.0           1           1.0
  record fetched      12570         0         70.7         37710         37710.0
```

The problem would only occur if the table had an index referencing it.

This problem would occur due to an error in the processing of erased rows. When a row was chosen to be erased, it was fetched for update access and incorrectly marked as "modified", even though no changes had actually been made to the row at that point in time. If the row was fragmented on disk, then the row would be reconstructed in memory using the fragments. Once the row was fetched, any corresponding index nodes would be fetched so that the index entries pointing to the row could be removed from the index nodes. Oracle Rdb can only have one current record at a time so the data row to be erased had to be freed before an index node could be updated. Any time a fragmented record is no longer the current record it must be restored on disk at that point in time. The fragmented data row was not actually erased yet so the original contents were written back to disk. If Oracle Rdb detects that space is available on the page used for the data row that would allow the fragmented row to become defragmented: it will delete the old fragment chain and restore the record on disk. Since the transaction was erasing rows from the database, it was highly likely that pages that were once full would have space become available, thus it was likely that the fragmented row could be restored on disk with the fragmentation removed. Deleting the previously fragmented row would cause the "record erased" counter to be incremented, even though no data rows had truly been erased at that point in time. After the current data row was restored to disk, the index nodes would be updated, then the data row would be made the current record again so that it could be erased. The data row would then actually be erased, causing the "record erased" counter to be incremented again.

The solution to this problem is to not mark the row as modified when it is first fetched. Only when the row is actually being erased will it be marked as modified. That prevents fragmented rows from getting restored on disk before the index nodes are updated.

This problem can be avoided by eliminating fragmented rows in the database.

This problem has been corrected in Oracle Rdb Release 7.1.4.3.

## **2.5.2 Some RMU/SHOW STATISTICS Screens Not Cleared Properly**

Bug 4755183

In several RMU/SHOW STATISTICS multi-page displays for file information, the final page of the display would not be correctly cleared after the last valid item on the page.

This problem has been corrected in Oracle Rdb Release 7.1.4.3. The display is now correctly cleared in the "File IO Overview" and "File Lock Overview" displays.

---

# **Chapter 3**

## **Software Errors Fixed in Oracle Rdb Release 7.1.4.2**

This chapter describes software errors that are fixed by Oracle Rdb Release 7.1.4.2.

## 3.1 Software Errors Fixed That Apply to All Interfaces

### 3.1.1 Bugcheck in PSII2REMOVEBOTTOM

Bug 4517441

In unusual cases, when removing entries from an index it was possible for Rdb to bugcheck with an exception in PSII2REMOVEBOTTOM. This problem was due to an incorrect pointer being used referencing index node content.

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

### 3.1.2 Logical RDM\$BIND\_RW\_TX\_CHECKPOINT\_ADVANCE Removed

Bug 1584167

Prior to Oracle Rdb Release 7.1.2, if the logical *RDM\$BIND\_RW\_TX\_CHECKPOINT\_ADVANCE* was not defined to be 1, read write transactions that did not make any database modifications would not advance their fast commit checkpoint location. In Release 7.1.2, in response to Bug 2439694, the checkpointing code was restructured such that checkpoints may advance at the end of any transaction, whether or not the transactions made any database modifications. That change made the logical *RDM\$BIND\_RW\_TX\_CHECKPOINT\_ADVANCE* no longer necessary. It has been removed.

### 3.1.3 Wrong Result from Constant View Column

Bug 4448141

The following query should find one row.

```
set flags 'strategy,detail';
select employee_id, i.const_col, j.const_col
from employees
  join mfp_v1 i using (employee_id)
  left outer join mfp_v2 j using (employee_id)
  where i.const_col = 9 and j.const_col is NULL
```

Tables:

```
0 = EMPLOYEES
1 = EMPLOYEES
2 = SALARY_HISTORY
3 = EMPLOYEES
4 = SALARY_HISTORY
```

Conjunct: (9 = 9) AND MISSING (9)

Match (Left Outer Join)

Outer loop

Conjunct: 0.EMPLOYEE\_ID = 1.EMPLOYEE\_ID

Match

Outer loop (zig-zag)

## Oracle® Rdb for OpenVMS

```
Index only retrieval of relation 0:EMPLOYEES
  Index name  EMP_EMPLOYEE_ID [0:0]
Inner loop
  Temporary relation
  Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation 1:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Cross block entry 2
    Conjunct: <agg0> <> 0
    Aggregate-F1: 0:COUNT-ANY (<subselect>)
    Index only retrieval of relation 2:SALARY_HISTORY
      Index name  SH_EMPLOYEE_ID [1:1]
      Keys: 1.EMPLOYEE_ID = 2.EMPLOYEE_ID
Inner loop
  Temporary relation
  Sort: 3.POSTAL_CODE(a)
  Conjunct: <agg1> <> 0
  Match      (Agg Outer Join)
  Outer loop
    Get      Retrieval by index of relation 3:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Inner loop      (zig-zag)
    Aggregate-F1: 1:COUNT-ANY (<subselect>)
    Index only retrieval of relation 4:SALARY_HISTORY
      Index name  SH_EMPLOYEE_ID [0:0]
0 rows selected
```

where the views mfp\_v1 and mfp\_v2 are defined as follows:

```
create view mfp_v1 (employee_id, const_col) as
select a.employee_id, 9 from employees a
  where a.employee_id = any (select z.employee_id from salary_history z )
;

create view mfp_v2 (employee_id, const_col) as
select a.postal_code, 9 from employees a
  where a.employee_id = any (select z.employee_id from salary_history z )
;
```

If the WHERE predicates are removed, the query finds 100 rows with the values 9 and NULL, as in the following example.

```
select employee_id, i.const_col, j.const_col
from employees
  join mfp_v1 i using (employee_id)
  left outer join mfp_v2 j using (employee_id);
```

Tables:

```
0 = EMPLOYEES
1 = EMPLOYEES
2 = SALARY_HISTORY
3 = EMPLOYEES
4 = SALARY_HISTORY
```

```
Match      (Left Outer Join)
Outer loop
  Conjunct: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
Match
  Outer loop      (zig-zag)
    Index only retrieval of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Inner loop
```

```

Temporary relation
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation 1:EMPLOYEES
    Index name  EMP_EMPLOYEE_ID [0:0]
  Cross block entry 2
    Conjunct: <agg0> <> 0
    Aggregate-F1: 0:COUNT-ANY (<subselect>)
    Index only retrieval of relation 2:SALARY_HISTORY
    Index name  SH_EMPLOYEE_ID [1:1]
    Keys: 1.EMPLOYEE_ID = 2.EMPLOYEE_ID
Inner loop
  Temporary relation
  Sort: 3.POSTAL_CODE(a)
  Conjunct: <agg1> <> 0
  Match   (Agg Outer Join)
  Outer loop
    Get   Retrieval by index of relation 3:EMPLOYEES
    Index name  EMP_EMPLOYEE_ID [0:0]
  Inner loop   (zig-zag)
    Aggregate-F1: 1:COUNT-ANY (<subselect>)
    Index only retrieval of relation 4:SALARY_HISTORY
    Index name  SH_EMPLOYEE_ID [0:0]
EMPLOYEE_ID  I.CONST_COL  J.CONST_COL
00164        9          NULL
00165        9          NULL
00166        9          NULL
00167        9          NULL
...etc...
00435        9          NULL
00471        9          NULL
100 rows selected

```

There is no known workaround for this problem.

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

1. The main query selects from two views left outer joined by the common column 'employee\_id' where both views contain the constant literal as one of the columns. In this case, the constant literal is '9'.
2. The filter predicates in the WHERE clause reference the constant column of each view.

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

### 3.1.4 Monitor Fails with VASFULL Error

Bug 4548613

The Oracle Rdb database monitor process could fail with a COSI-F-VASFULL error after processing a large number of blocking ASTs for cluster membership (MEMBIT) locks. The monitor logfile may contain entries similar to the following:

```

10-AUG-2005 23:47:22.89 - received request from remote node to join
- database name "DEV:[DIR]FILE.RDB;1"
- starting forced-exit shutdown of database:
- "%COSI-F-VASFULL, virtual address space full"

```

A bugcheck dump may be generated containing an exception similar to the following:

## Oracle® Rdb for OpenVMS

```
***** Exception at 000D7E78 : MONLCK$GET_UMAIL + 00000058
%COSI-F-VASFULL, virtual address space full
```

The cluster membership lock is receiving a blocking AST when messages similar to the following are seen in the monitor logfile:

```
10-AUG-2005 00:02:37.42 - received request from remote node to join
  - database name "DEV:[DIR]FILE.RDB;1"
  - cluster watcher waiting for MEMBIT lock

10-AUG-2005 00:02:37.42 - lock granted to cluster watcher
  - database name "DEV:[DIR]FILE.RDB;1"
  - cluster watcher is active

10-AUG-2005 00:02:56.24 - received notification of remote monitor deaccess
  - database name "DEV:[DIR]FILE.RDB;1"
  - monitor 2 (OTRNOD) is shutdown
  - cluster recovery completed successfully
  - cluster watcher is active
```

This problem can be avoided by manually opening databases that are often accessed within a cluster by using the RMU/OPEN command.

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

### 3.1.5 Wrong Result from UNION View Query with NOT STARTING WITH Clause

Bugs 4550715 and 4327112

The following query should select two rows from the FW\_V view for rows where column F3 does not start with 'Z'. As indicated next to the rows returned, the results are wrong. This is due to an incorrect query strategy, as will be explained farther on.

```
set flags 'strategy,detail';
SELECT * FROM FW_V WHERE F3 NOT STARTING WITH 'Z';
Tables:
  0 = TAB_A
  1 = TAB_B
  2 = TAB_B
  3 = TAB_C
Reduce: <mapped field>, <mapped field>, <mapped field>
Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
Merge of 2 entries
  Merge block entry 1
  Cross block of 3 entries
    Cross block entry 1
      Get Retrieval sequentially of relation 0:TAB_A
    Cross block entry 2
      Aggregate: 0:VIA (1.FLD3)
      Conjunct: 1.FLD2 = 0.FLD2
      Get Retrieval sequentially of relation 1:TAB_B
    Cross block entry 3
      Aggregate: 1:VIA (2.FLD3)
      Conjunct: 2.FLD2 = 0.FLD2
      Get Retrieval sequentially of relation 2:TAB_B
  Merge block entry 2
```

## Oracle® Rdb for OpenVMS

```
Conjunct: NOT (3.F3 STARTING WITH 'Z')
Get      Retrieval sequentially of relation 3:TAB_C
FLD1    FLD2    F3
1       A       Z      <== wrong
2       A       Z      <== wrong
3       B       1      <== correct
4       C       Z      <== wrong
aaa     bbb     ccc     <== correct
5 rows selected
```

The view, FW\_V, is defined below. Following the view definition is a SELECT query similar to the problem query shown earlier but without the clause "WHERE F3 NOT STARTING WITH 'Z'".

```
CREATE VIEW FW_V AS
SELECT
    FLD1,
    A.FLD2,
    NVL((SELECT FLD3 FROM TAB_B B WHERE B.FLD2 = A.FLD2), 'Z')
    FROM TAB_A A
UNION
SELECT F1, F2, F3
    FROM TAB_C;
```

```
select * from fw_v;
FLD1    FLD2    F3
1       A       Z
2       A       Z
3       B       1
4       C       Z
aaa     bbb     ccc
5 rows selected
```

The query returns the correct result if the NOT operator is removed and the reason is because the conjunct "STARTING WITH 'Z'" appears correctly in the detailed strategy, as seen in the following example.

```
SELECT * FROM FW_V WHERE F3 STARTING WITH 'Z';
Tables:
  0 = TAB_A
  1 = TAB_B
  2 = TAB_B
  3 = TAB_C
Reduce: <mapped field>, <mapped field>, <mapped field>
Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
Conjunct: <mapped field> STARTING WITH 'Z'      <=== This conjunct appears here
Merge of 2 entries
  Merge block entry 1
  Cross block of 3 entries
    Cross block entry 1
      Get      Retrieval sequentially of relation 0:TAB_A
    Cross block entry 2
      Aggregate: 0:VIA (1.FLD3)
      Conjunct: 1.FLD2 = 0.FLD2
      Get      Retrieval sequentially of relation 1:TAB_B
    Cross block entry 3
      Aggregate: 1:VIA (2.FLD3)
      Conjunct: 2.FLD2 = 0.FLD2
      Get      Retrieval sequentially of relation 2:TAB_B
  Merge block entry 2
  Conjunct: 3.F3 STARTING WITH 'Z'
  Get      Retrieval sequentially of relation 3:TAB_C
```

```

FLD1   FLD2   F3
1      A      Z
2      A      Z
4      C      Z
3 rows selected

```

If we compare the strategies of the good and bad queries, the only difference is that the bad query was missing the conjunct "NOT STARTING WITH", but the second good query (the one without the NOT operator), correctly places the conjunct "STARTING WITH 'Z'" outside the "Merge" entries.

There is no known workaround for this problem.

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

1. The main query selects from the derived table of a UNION clause with a filter predicate.
2. The first leg of the UNION clause contains a select query with a NVL function on a subselect query.
3. The second leg of the UNION clause contains a select query from a table.
4. The WHERE clause contains the NOT operator in front of a string function (STARTING WITH).

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

### 3.1.6 Problem with Partitioned Indexes and Descending Segments with Starting With and Like

Bug 4536197

In prior releases of Oracle Rdb, a problem with partitioned indexes containing descending index segments could cause some queries to fail to deliver a full set of resultant records.

During retrieval of records using the index, some partitions belonging to the index may be overlooked and subsequently those records that matched the search criteria in the overlooked partitions would not be returned by the query.

This problem only occurred when either the 'starting with' or 'like' operator was used in the query.

```

SQL> create database file testp
cont>
cont>     create storage area rdb$system
cont>     page format is uniform
cont>
cont>     create storage area uniform1
cont>     filename uniform1
cont>     page format is uniform
cont>
cont>     create storage area uniform2
cont>     filename uniform2
cont>     page format is uniform;
SQL>
SQL> create table employees (employee_id char(5));
SQL>
SQL>
SQL> !
SQL> !   Create data values for inserts into records such that
SQL> !   each partition in the index (once it is created) will

```

## Oracle® Rdb for OpenVMS

```
SQL> !   have four data records
SQL> !
SQL> insert into employees (employee_id) values ('00100');
1 row inserted
SQL> insert into employees (employee_id) values ('00199');
1 row inserted
SQL>
SQL> insert into employees (employee_id) values ('00900');
1 row inserted
SQL> insert into employees (employee_id) values ('00999');
1 row inserted
SQL> commit;
SQL>
SQL> create unique index test_index on EMPLOYEES ( EMPLOYEE_ID desc)
cont>   type is SORTED
cont>   disable compression
cont>   store
cont>     using (EMPLOYEE_ID)
cont>     in uniform1
cont>     with limit of ('00900')
cont>     otherwise in uniform2;
SQL>
SQL> select * from employees;
  EMPLOYEE_ID
 00999
 00900
 00199
 00100
4 rows selected
SQL> !
SQL> ! first some queries that work
SQL> !
SQL> select * from employees where employee_id like '001%';
  EMPLOYEE_ID
 00199
 00100
2 rows selected
SQL> select * from employees where employee_id like '009%';
  EMPLOYEE_ID
 00999
 00900
2 rows selected
SQL> select * from employees where employee_id starting with '001';
  EMPLOYEE_ID
 00199
 00100
2 rows selected
SQL> !
SQL> ! now some that fail
SQL> !
SQL> select * from employees where employee_id starting with '009';
0 rows selected
SQL> select * from employees where employee_id like '%';
  EMPLOYEE_ID
 00199
 00100
2 rows selected
SQL> rollback;
```

A workaround for this problem is to not use descending segments in the partitioned index.

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

### 3.1.7 AIJ Server Process Not Always Run Under RDMAIJ Account

Bug 2379799

The RDMAIJ (AIJ Server) process would not always run under the RDMAIJ account if a proxy was defined for the username that started the Oracle Rdb monitor on the master node. For example, consider the following Hot Standby configuration:

- A privileged account on the master node is used to start the Oracle Rdb monitor process.
- An identically named user account also exists on the standby node.
- A proxy has been defined on the standby node for the privileged account on the master node that allows proxy network access to the standby node from the master node.

In a configuration like the one described above, it was possible for the AIJ Server process to run under the username of the privileged account instead of under the RDMAIJ username declared in the network object definition. Thus the AIJ Server process would inherit the default privileges defined for the privileged account instead of the privileges defined for the RDMAIJ account.

Typically this would not cause any problems except in situations where the default privileges for the user were less than the RDMAIJ process. In that situation, the AIJ Server process would not be able to perform some privileged functions that it would have been able to perform if it was running under the RDMAIJ username. For example, the AIJ Server process might not be able to increase its process priority, or it might not be able to determine if the Log Recovery Server (LRS) was accumulating CPU.

This problem can be corrected by changing the DECnet network object to disallow proxy access. For DECnet phase IV nodes, the following NCP command can be used:

```
$ MCR NCP SET OBJECT RDMAIJ{nn} PROXY NONE
$ MCR NCP DEFINE OBJECT RDMAIJ{nn} PROXY NONE
```

For DECnet-plus nodes the following NCL command can be used:

```
$ MCR NCL SET NODE 0 SESSION CONTROL APPLICATION RDMAIJ{nn} -
  INCOMING PROXY = FALSE
```

Note that in the above examples the "{nn}" should be replaced with the appropriate version number. For example, the object name might be "RDMAIJ71".

This problem has been corrected in Oracle Rdb Release 7.1.4.2. The *RDMAIJSERVER{nn}\_NCP.COM* and *RDMAIJSERVER{nn}\_NCL.COM* files have been updated to disable proxy access to the RDMAIJ network object.

### 3.1.8 ALS Would Not Stay Stopped

Bug 3297321

If the AIJ Log Server (ALS) mode was set to "AUTOMATIC" and the ALS was manually stopped using the *RMU/SERVER AFTER\_JOURNAL STOP* command, the ALS would automatically restart when another user attached to the database. The ALS should not have been restarted.

This problem has been corrected in Oracle Rdb Release 7.1.4.2. The ALS process will not restart unless it is manually restarted.

### 3.1.9 Bugchecks at PSIHASHBKT\$INSERT\_INTO\_SYS + 00000068

Bug 4587490

In high contention environments where multiple users were inserting hashed index entries into the same mixed format storage area, it was possible for a bugcheck to occur with the following exception:

```
***** Exception at 014719A8 : PSIHASHBKT$INSERT_INTO_SYS + 00000068
%COSI-F-BUGCHECK, internal consistency failure
```

This bugcheck would result after the following sequence of events:

1. Oracle Rdb would attempt to insert an index entry into a page.
2. The system record on the page would not contain a hash bucket pointer for the desired index.
3. The inserting process would create a hash bucket for the index. While doing so it would release its lock on the system record.
4. Another process would also attempt to insert an index entry on the same page, find that the system record had no entry for the index, and also create a hash bucket for the same index.
5. One of the inserting processes would then insert a pointer in the system record for the newly created hash bucket.
6. The other inserting process would refetch the system record, check to make sure that it still did not have any entry for the index, discover that an entry exists, and bugcheck.

This problem has been corrected in Oracle Rdb Release 7.1.4.2. The lock on the system record will be retained while the hash bucket is being created, preventing other processes from updating the system record at the same time.

### 3.1.10 Bugchecks or Corruption in Indexes of TYPE IS SORTED RANKED

Bug 4437323

During a sequence of inserts on a table with an index of *TYPE IS SORTED RANKED*, it was possible that the index could become corrupt. This could cause the index to be left corrupt or could result in a bugcheck dump if subsequent delete operations were performed in the same transaction.

The problem is quite rare and requires that a series of at least three consecutive inserts occur on exactly the same duplicate node in a specific pattern.

If the index was corrupted, a subsequent verify would produce output similar to the following example.

## Oracle® Rdb for OpenVMS

```
SQL> insert into t1 values (0,33);
1 row inserted
SQL> insert into t1 values (0,11);
1 row inserted
SQL> insert into t1 values (0,22);
1 row inserted
SQL> commit;
SQL> Exit
$ rmu/verify/index/data test
%RMU-W-IDXDATMIS, Index I1 does not point to a row in table T1.
      Logical dbkey of the missing row is 68:12:1.
%RMU-W-BADIDXREL, Index I1 either points to a non-existent record or
      has multiple pointers to a record in table T1.
      The logical dbkey in the index is 68:14:1.
```

Various bugchecks may result depending on how the corruption was detected. The following shows two different examples of the call frames in bugcheck dumps from this problem.

```
***** Exception at 012D7DC8 : PIOFETCH$WITHIN_DB + 000005A8
Saved PC = 012D5414 : PIOFETCH$FETCH + 000002C4
Saved PC = 012D44D4 : PIO$FETCH + 00000914
Saved PC = 01220044 : DIOFETCH$FETCH_VISIBLE_SEG + 00000154
Saved PC = 01221414 : DIOFETCH$FETCH_ONE_LINE + 00000AB4
Saved PC = 01221978 : DIO$FETCH_DBKEY + 000002F8
Saved PC = 01044848 : RDMS$$KOD_FIND_CURRENT_REC + 00000558
Saved PC = 01023684 : RDMS$$EXE_NEXT + 00000954
```

```
***** Exception at 0141D9DC : PSII2REMOVEDUPBBC + 0000118C
Saved PC = 0141A3CC : PSII2REMOVEBOTTOM + 0000071C
Saved PC = 01413E74 : PSII2REMOVET + 000001F4
Saved PC = 014140B4 : PSII2REMOVET + 00000434
Saved PC = 014140B4 : PSII2REMOVET + 00000434
Saved PC = 014140B4 : PSII2REMOVET + 00000434
Saved PC = 0141481C : PSII2REMOVETREE + 000001FC
Saved PC = 016DA80C : RDMS$$KOD_REMOVE_TREE + 0000162C
Saved PC = 016A29DC : RDMS$$EXE_ACTION + 0000321C
```

If the bugcheck occurred during the transaction that corrupted the index, the transaction would be rolled back and the index would not be left corrupt.

If the problem did not cause a bugcheck during the offending transaction, the transaction could commit and the index would be left corrupt. The index should be verified using the */INDEX/DATA* qualifiers of RMU Verify. If errors are reported, the index should be dropped and recreated to eliminate the corruption.

The problem can be avoided by adding columns to the index to make it unique, or by using alternate index types.

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

### 3.1.11 Unexpected Bugcheck when Formatting Illegal Date/Time Value

Bug 4557990

In previous versions of Oracle Rdb, it was possible that an invalid value in a TIME, DATE (ansi), or

TIMESTAMP would result in a reported ACCVIO or a bugcheck from Rdb. This might occur when using Interactive SQL to display the value or when using RMU/UNLOAD to format the value as text.

This problem has been corrected in Oracle Rdb Release 7.1.4.2. Oracle Rdb now uses a field of "\*\*\*\*" to indicate the illegal date/time data.

### 3.1.12 Wrong Result by Query with Constant Column Defined in a View

Bugs 1752645 and 4155086

The following query should return 100 rows with NULL for the column "v.const\_col".

```
create view mfp_view (employee_id, const_col) as
select e.postal_code, 1 from employees e
  where e.employee_id = any (select z.employee_id from salary_history z );

select employee_id, v.const_col
  from employees left outer join mfp_view v using (employee_id)
 where v.const_col is null;
Tables:
  0 = EMPLOYEES
  1 = EMPLOYEES
  2 = SALARY_HISTORY
Conjunct: MISSING (1)
Match    (Left Outer Join)
Outer loop
  Index only retrieval of relation 0:EMPLOYEES
    Index name  EMP_EMPLOYEE_ID [0:0]
Inner loop
  Temporary relation
  Sort: 1.POSTAL_CODE(a)
  Conjunct: <agg0> <> 0
  Match    (Agg Outer Join)
    Outer loop
      Get      Retrieval by index of relation 1:EMPLOYEES
        Index name  EMP_EMPLOYEE_ID [0:0]
      Inner loop  (zig-zag)
        Aggregate-F1: 0:COUNT-ANY (<subselect>)
        Index only retrieval of relation 2:SALARY_HISTORY
          Index name  SH_EMPLOYEE_ID [0:0]
0 rows selected
```

This problem occurs when the main query is an outer join between the EMPLOYEES table and a view using the EMPLOYEE\_ID as the join predicate, where the view contains a column of either constant value or an expression of all constant values.

```
create view mfp_view (employee_id, const_col) as
select e.postal_code, 1+2 from employees e
  where e.employee_id = any (select z.employee_id from salary_history z );

select employee_id, v.const_col
  from employees left outer join mfp_view v using (employee_id)
 where v.const_col is null;
Tables:
  0 = EMPLOYEES
  1 = EMPLOYEES
```

```

2 = SALARY_HISTORY
Conjunct: MISSING (1 + 2)
Match    (Left Outer Join)
Outer loop
  Index only retrieval of relation 0:EMPLOYEES
    Index name  EMP_EMPLOYEE_ID [0:0]
Inner loop
  Temporary relation
  Sort: 1.POSTAL_CODE(a)
  Conjunct: <agg0> <> 0
  Match    (Agg Outer Join)
    Outer loop
      Get      Retrieval by index of relation 1:EMPLOYEES
        Index name  EMP_EMPLOYEE_ID [0:0]
      Inner loop  (zig-zag)
        Aggregate-F1: 0:COUNT-ANY (<subselect>)
        Index only retrieval of relation 2:SALARY_HISTORY
          Index name  SH_EMPLOYEE_ID [0:0]
0 rows selected

```

The similar query works if the constant column in the view is concatenated with another column of the EMPLOYEES table, for example middle\_initial.

```

create view mfp_view (employee_id, const_col) as
select e.postal_code, '1' || e.middle_initial from employees e
  where e.employee_id = any (select z.employee_id from salary_history z );

select employee_id, v.const_col
  from employees left outer join mfp_view v using (employee_id)
 where v.const_col is null;
Tables:
  0 = EMPLOYEES
  1 = EMPLOYEES
  2 = SALARY_HISTORY
Conjunct: MISSING ('1' || 1.MIDDLE_INITIAL)
Match    (Left Outer Join)
Outer loop
  Index only retrieval of relation 0:EMPLOYEES
    Index name  EMP_EMPLOYEE_ID [0:0]
Inner loop
  Temporary relation
  Sort: 1.POSTAL_CODE(a)
  Conjunct: <agg0> <> 0
  Match    (Agg Outer Join)
    Outer loop
      Get      Retrieval by index of relation 1:EMPLOYEES
        Index name  EMP_EMPLOYEE_ID [0:0]
      Inner loop  (zig-zag)
        Aggregate-F1: 0:COUNT-ANY (<subselect>)
        Index only retrieval of relation 2:SALARY_HISTORY
          Index name  SH_EMPLOYEE_ID [0:0]
EMPLOYEE_ID  I.CONST_COL
00164        NULL
00165        NULL
...etc...
00471        NULL
100 rows selected

```

The fix for this bug was backed out to fix the problem in Bug 4155086. It is now redesigned with the correct solution which also correctly handles the case where a view constant column of the same value exists in more

than one view.

For example, the following example shows that a view constant of the value '9' exists in two different views.

```
attach 'file mf_personnel';

create view mfp_v1 (employee_id, const_col) as
select a.employee_id, 9 from employees a
  where a.employee_id = any (select z.employee_id from salary_history z )
  group by a.employee_id;

create view mfp_v2 (employee_id, const_col) as
select a.postal_code, 9 from employees a
  where a.employee_id = any (select z.employee_id from salary_history z )
  group by a.postal_code;
```

The following query should find 100 rows with values 9 and NULL.

```
select count(*)
from employees
  join mfp_v1 i using (employee_id)
  left outer join mfp_v2 j using (employee_id)
  where i.const_col = 9 and j.const_col is NULL
  ;
Aggregate      Conjunct
Match      (Left Outer Join)
Outer loop
  Conjunct
  Match
    Outer loop
      Reduce
      Cross block of 2 entries
      Cross block entry 1
        Index only retrieval of relation EMPLOYEES
        Index name  EMP_EMPLOYEE_ID [0:0]
      Cross block entry 2
        Conjunct      Aggregate-F1
        Index only retrieval of relation SALARY_HISTORY
        Index name  SH_EMPLOYEE_ID [1:1]
    Inner loop      (zig-zag)
      Index only retrieval of relation EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Inner loop
    Temporary relation      Reduce Sort      Conjunct
    Match      (Agg Outer Join)
    Outer loop
      Get      Retrieval by index of relation EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
    Inner loop      (zig-zag)
      Aggregate-F1      Index only retrieval of relation SALARY_HISTORY
      Index name  SH_EMPLOYEE_ID [0:0]

0
1 row selected
```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

### 3.1.13 Various Errors Using Read Only Areas with Global Buffers

Bug 4630467

Various problems could occur when accessing read only storage areas when global buffers were enabled and multiple processes were accessing the same pages at the same time. For example, bugchecks with exceptions similar to the following could occur:

```
***** Exception at 00EA6948 : RDMS$$EXE_NEXT + 000009D8
%COSI-F-BUGCHECK, internal consistency failure
```

This problem would occur because the buffer was not properly interlocked, allowing multiple users to read data into the buffer at the same time. The contents of the buffer would not be consistent until all I/Os had completed. For very brief instances, the buffer could contain some zeroes while it was being filled in by the I/O request. Some users could be accessing the buffer while it contained the transient zeroes, leading to various failures or incorrect results.

This problem can be avoided by changing the storage area to be READ WRITE or by disabling global buffers.

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

### 3.1.14 Many Processes Stalling with "hibernating on AIJ submission"

Bug 4638908

If a database was open on multiple nodes, and the AIJ Log Server (ALS) process on one of the nodes failed, the ALS on the node that recovered the failed ALS would start running slowly, only processing AIJ writes every five seconds. This would result in users stalling for an unusually long time with the stall message "hibernating on AIJ submission".

This problem was introduced in Oracle Rdb Release 7.1.4.1. In that release, the Database Recovery process (DBR) was changed to clear global data structures related to the ALS when recovering a failed ALS. However, it would incorrectly clear the local node's structures when recovering a failed ALS from another node when it should have only cleared those structures when the local ALS failed.

This problem can be avoided by not using the /ABORT=DELPRC option when closing a database that has an ALS running. Or it can be avoided by manually stopping the ALS prior to forcing other users out of the database with /ABORT=DELPRC. The ALS can be manually stopped by issuing the RMU/SERVER AFTER STOP command.

If an ALS process starts running slowly, the problem can be resolved by manually stopping and restarting the ALS. For example:

```
$ RMU/SERVER AFTER STOP MF_PERSONNEL
$ RMU/SERVER AFTER START MF_PERSONNEL
```

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

### 3.1.15 DBR Bugchecks at PIO\$BIND + 000003B8

Bug 4661618

The database recovery process (DBR) could fail with a bugcheck similar to the following when the Row Cache feature was being used in conjunction with Page Transfer Via Memory:

```
***** Exception at 00188958 : PIO$BIND + 000003B8
%COSI-F-BUGCHECK, internal consistency failure
```

The failure would occur when the DBR was attempting to do a "node failure recover all" recovery, but it was not allocated any global buffers by the monitor.

A "node failure" is considered to have occurred when all users from a node stop accessing a database without clearing their user entries in the database root file, and the monitor accessing the database ends its access without clearing its member ID (MEMBIT) entry in the database root file. When a database monitor detects that a node failure has occurred, special recovery algorithms are invoked if the Row Cache or Page Transfer Via Memory features are being used in the database. In that situation, the DBR will recover all failed users in the database in one pass, instead of having separate DBRs started for each failed user. This is referred to as the "recover all" algorithm. If a recover all recovery is being done, the monitor will temporarily disable global buffers so that the DBR process is not constrained by the currently configured global buffer user limit. The DBR is expected to allocate local buffers for page I/O.

In the reported problem, there was a sequence of events that could cause the monitor to not disable global buffers and not allocate any global buffers to the DBR process when the DBR attempted to do node failure recovery. For this problem to occur, the database had to have the following features enabled:

- Global Buffers
- Row Cache
- Page Transfer Via Memory
- AIJ Log Server (ALS) automatic

When the following sequence of events occurred, the DBR would fail because it did not have any buffers allocated.

1. The database was manually opened with RMU/OPEN.
2. The ALS started and attached to the database.
3. The Row Cache Server (RCS) started.
4. An RMU/CLOSE command was issued.
5. At the same time that the RMU/CLOSE was issued, the RCS attempted to attach to the database. The attach request was refused by the monitor because the database was being closed.
6. The RCS failed because the monitor refused its attach request.
7. The monitor detected that an RCS process failed, which requires an immediate database shutdown.
8. The monitor terminated the ALS process using \$DELPRC.
9. The database is then completely shutdown, with an ALS process needing to be recovered.
10. The database is again opened with RMU/OPEN.
11. The monitor "cluster watcher" detected that the database required node failure recovery.
12. Node failure recovery detected that row cache was enabled and searched for a failed RCS process.
13. No failed RCS process was found.
14. At that point, since no failed RCS process was found, the monitor should have also checked to see if page transfer via memory was enabled. Page transfer via memory requires that the DBR use the same

node failure recovery algorithms used by row cache. However, the monitor did not check for page transfer via memory and instead treated the recovery as a regular recovery, not a "recover all" recovery. Since it was a regular recovery, the monitor did not disable global buffers.

15. The DBR started and sent an attach request to the monitor. Since it was doing a recover all recovery and global buffers were enabled, the DBR expected that it would have buffers allocated to it by the monitor.
16. The monitor saw that the DBR was recovering an ALS process. The ALS does not use any page buffers so the monitor does not allocate any buffers to the DBR.
17. The DBR finds that it was not allocated any buffers and fails since it needs buffers to do a recover all recovery.

After the above occurred, any attempt to open the database would fail. The DBR would continue to terminate with the bugcheck described. The database had to be restored and the after image journals applied to recover it to its last consistent state.

This problem has been corrected in Oracle Rdb Release 7.1.4.2. The monitor will now always check for page transfer via memory when determining if a recover all recovery is necessary.

### 3.1.16 Recovered Database Missing Updates

Bug 4667857

If a database was manually recovered using the original (not backed up) journals and the journals had been renamed or copied from their original location, or the database was restored with the /NEW option and instead of listing all journals in a single recover command separate recover commands were used for each journal, then the resulting database could be missing updates.

The problem can be demonstrated with the following example.

```
$ SQL$
CREATE DATABASE FILENAME TESTDB
  CREATE STORAGE AREA RDB$SYSTEM FILENAME TESTDB
  CREATE TABLE T1 (C1 INT, C2 CHAR (900))
  CREATE STORAGE MAP SM1 FOR T1
  DISABLE COMPRESSION;

DISCONNECT ALL;

ALTER DATABASE FILE TESTDB
  RESERVE 1 JOURNAL
  JOURNAL IS ENABLED
  ADD JOURNAL JOURNAL1 FILENAME 'SYS$DISK:[ ]JOURNAL1.AIJ'
  ALLOCATION IS 512
  ADD JOURNAL JOURNAL2 FILENAME 'SYS$DISK:[ ]JOURNAL2.AIJ'
  ALLOCATION IS 512;
%RDMS-W-DOFULLBCK, full database backup should be done to ensure future recovery
%RDMS-W-DOFULLBCK, full database backup should be done to ensure future recovery

EXIT;
$
$ RMU/BACKUP/NOLOG TESTDB TESTDB
$
$ SQL$
  ATTACH 'FILENAME TESTDB';
  INSERT INTO T1 VALUES (1, 'First transaction');
```

## Oracle® Rdb for OpenVMS

```
1 row inserted
COMMIT;

BEGIN
DECLARE :INSERT_COUNT INT;

-- Insert enough data to fill the first journal and overflow into
-- the second.

WHILE :INSERT_COUNT < 399
LOOP
  INSERT INTO T1 VALUES (:INSERT_COUNT, 'Second transaction');
  SET :INSERT_COUNT = :INSERT_COUNT + 1;
END LOOP;
END;

-- Note how many rows were inserted.

SELECT COUNT (*) FROM T1;

          400
1 row selected

-- Commit second transaction. Part of the transaction should be
-- in the first journal; the rest should end up in the second.

COMMIT;

EXIT;
$
$ SQL$
DROP DATABASE FILENAME TESTDB;
EXIT;
$
$ ! Rename the journals to prevent automatic recovery.
$
$ RENAME JOURNAL1.AIJ COPY1.AIJ
$ RENAME JOURNAL2.AIJ COPY2.AIJ
$
$ ! Restore the db.
$
$ RMU/RESTORE/NOLOG/NOCCD/NORECOVER TESTDB
%RMU-I-AIJRSTAVL, 0 after-image journals available for use
%RMU-I-AIJISOFF, after-image journaling has been disabled
$
$ ! Recover first journal. It should end saying
$ ! %RMU-I-LOGRECSTAT, transaction with TSN n:nnn is active
$ ! ...
$ ! %RMU-I-LOGSUMMARY, total 1 transaction rolled back
$
$ RMU/RECOVER/LOG/NOTRACE COPY1
%RMU-I-LOGRECDB, recovering database file DEV:[DIR]TESTDB.RDB;1
%RMU-I-LOGOPNAIJ, opened journal file DEV:[DIR]COPY1.AIJ;1 at 11-OCT-2005
17:27:30.43
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-LOGRECOVR, 2 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJACTIVE, 1 active transaction not yet committed or aborted
%RMU-I-LOGRECSTAT, transaction with TSN 0:129 is active
%RMU-I-AIJSUCCE, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence number
```

## Oracle® Rdb for OpenVMS

```
needed will be 1
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-LOGSUMMARY, total 2 transactions committed
%RMU-I-LOGSUMMARY, total 1 transaction rolled back
%RMU-I-LOGSUMMARY, total 0 transactions ignored
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence number
needed will be 1
%RMU-I-AIJNOENABLED, after-image journaling has not yet been enabled
$
$ ! The following should fail with
$ ! %RMU-F-AIJNORCVR, recovery of this journal must start with sequence 0
$ ! but in the BUG report it did not.
$
$ RMU/RECOVER/LOG/NOTRACE COPY2
%RMU-I-LOGRECDB, recovering database file DEV:[DIR]TESTDB.RDB;1
%RMU-I-LOGOPNAIJ, opened journal file DEV:[DIR]COPY2.AIJ;1 at 11-OCT-2005
17:27:30.48
%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed
%RMU-I-LOGRECOVR, 1 transaction committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence number
needed will be 2
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-LOGSUMMARY, total 1 transaction committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
%RMU-I-LOGSUMMARY, total 0 transactions ignored
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence number
needed will be 2
%RMU-I-AIJNOENABLED, after-image journaling has not yet been enabled
$
$ SQL$
ATTACH 'FILENAME TESTDB';

-- Note how many rows were recovered.

SELECT COUNT (*) FROM T1;

        151
1 row selected
COMMIT;
EXIT;
$
```

Note that in the above example, the original database contained 400 entries but the recovered database only contained 151 entries.

This problem was caused by the RMU/RECOVER command mistakenly assuming that the copied journal would not have any subsequent journals in the sequence. After recovering the copied journal, the database was incorrectly marked as being consistent through the end of the last journal recovered, even if there were uncommitted transactions still active after the journal was processed.

This problem can be avoided by explicitly naming all journals in a single RMU/RECOVER command. For example:

```
$ RMU/RECOVER/LOG/NOTRACE COPY1, COPY2
```

### 3.1.16 Recovered Database Missing Updates

This problem has been corrected in Oracle Rdb Release 7.1.4.2. The database will no longer have the AIJ sequence number advanced if there are transactions still active after recovering a journal that is no longer in its original location.

### 3.1.17 Performance Improvement for Query with Constant Boolean Predicates

Bug 4205719

A customer reported that the following query, where the boolean condition always returned a known value of FALSE, used a full sequential retrieval and became very slow on a large table.

```
set flags 'strategy,detail';
select * from resumes where 1 = 2;
Tables:
  0 = RESUMES
Conjunct: 1 = 2
Get      Retrieval sequentially of relation 0:RESUMES
0 row selected
```

Although the condition was always false and 0 rows were returned, Rdb still performed a sequential table scan. In a database with about 1 million rows, this unnecessary table scan takes a lot of time.

Oracle Rdb has been changed to detect expressions of the following forms and to avoid doing index and table scans if those expressions are invariable and evaluated as false.

```
WHERE constant-expression
WHERE other-expression AND constant-expression
WHERE constant-expression AND other-expression
```

For example:

```
WHERE 1 = 2
WHERE (1 = 2) AND (LAST_NAME > '')
WHERE (LAST_NAME > '') AND (1 = 2)
```

This does not include expressions that contain host variables, as in the examples below, because host variables are considered to be variable.

```
WHERE :HV = 1
WHERE (:HV1 = 1) AND (LAST_NAME > '')
```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

### 3.1.18 Various Problems When LOCK PARTITIONING IS ENABLED

Bug 4675081

In prior releases of Oracle Rdb, various problems could be experienced when using the feature "LOCK PARTITIONING IS ENABLED". Problems may include excessive snapshot file growth, database corruption and instability.

This problem has been corrected in Oracle Rdb Release 7.1.4.2. Root area lock names are now correctly qualified with the storage area identified and the locks are correctly protected from having invalid lock value blocks by using system–owned null mode locks on each cluster node.

### 3.1.19 Wrong Result from Query with Zigzag Strategy

Bug 4616754

The following query, using zigzag strategy on both inner and outer legs, returns the wrong result (should return 3 rows).

```
SET FLAGS 'STRATEGY,DETAIL';

SELECT T2.C1, T2.C2, T2.C3
  FROM T2 JOIN T1
    ON T1.C1 = T2.C1 AND T1.C3 = T2.C3
 WHERE T1.C3 = 1 ;
Tables:
  0 = T2
  1 = T1
Conjunct: (1.C1 = 0.C1) AND (1.C3 = 0.C3)
Match
Outer loop      (zig-zag)
  Conjunct: 1.C3 = 1
  Index only retrieval of relation 1:T1
    Index name  T1_NDX [0:0]
Inner loop      (zig-zag)
  Conjunct: 0.C3 = 1
  Index only retrieval of relation 0:T2
    Index name  T2_NDX [0:0]
T2.C1      T2.C2      T2.C3
10          2          1
30          5          1
2 rows selected
```

The following are the rows in the outer loop:

```
SELECT C1,C3 FROM T1;
Index only retrieval of relation 0:T1
  Index name  T1_NDX [0:0]
C1           C3
10           1   <= 1st row
20           1   <= 2nd row
30           1   <= 3rd row
3 rows selected
```

The following are the rows in the inner loop with the comments in the parenthesis indicating a match or no match:

```
SELECT C1,C2,C3 FROM T2;
Index only retrieval of relation 0:T2
  Index name  T2_NDX [0:0]
C1           C2           C3
```

## Oracle® Rdb for OpenVMS

```

10          2          1  <= 1st row (match with 1st row from outer loop)
12          3          1  <= 2nd row (no match)
20          1          1  <= 3rd row (match with 2nd row from outer loop)
30          5          1  <= 4th row (match with 3rd row from outer loop)
4 rows selected

```

If the second row in the outer loop is deleted, the query works.

```

DELETE FROM T2 WHERE C1=12;

SELECT T2.C1, T2.C2, T2.C3
FROM T2 JOIN T1
  ON T1.C1 = T2.C1 AND T1.C3 = T2.C3
WHERE T1.C3 = 1 ;

Tables:
  0 = T2
  1 = T1
Conjunct: (1.C1 = 0.C1) AND (1.C3 = 0.C3)
Match
  Outer loop      (zig-zag)
    Conjunct: 1.C3 = 1
    Index only retrieval of relation 1:T1
      Index name  T1_NDX [0:0]
  Inner loop      (zig-zag)
    Conjunct: 0.C3 = 1
    Index only retrieval of relation 0:T2
      Index name  T2_NDX [0:0]
T2.C1      T2.C2      T2.C3
10          2          1
20          1          1
30          5          1
3 rows selected

```

The query also works if the value of the second column C2 of the first row in table T2 is updated to the same value as that of the third row, as in the following example.

```

UPDATE T2 SET C2 = 1 WHERE C1 = 10;

SELECT C1,C2,C3 FROM T2;
Index only retrieval of relation 0:T2
  Index name  T2_NDX [0:0]
C1          C2          C3
10          1          1  <= C2 is now equal to C2 of 3rd row
12          3          1
20          1          1  <= 3rd row
30          5          1
4 rows selected

SELECT T2.C1, T2.C2, T2.C3
FROM T2 JOIN T1
  ON T1.C1 = T2.C1 AND T1.C3 = T2.C3
WHERE T1.C3 = 1 ;

Tables:
  0 = T2
  1 = T1
Conjunct: (1.C1 = 0.C1) AND (1.C3 = 0.C3)
Match
  Outer loop      (zig-zag)
    Conjunct: 1.C3 = 1
    Index only retrieval of relation 1:T1
      Index name  T1_NDX [0:0]

```

```

Inner loop      (zig-zag)
  Conjunct: 0.C3 = 1
  Index only retrieval of relation 0:T2
    Index name  T2_NDX [0:0]
  T2.C1      T2.C2      T2.C3
  10          1          1
  20          1          1
  30          5          1
3 rows selected

```

This proves that the zigzag match strategy may work well with a set of data rows but may not work with a different set of data rows. This is the reason why this problem has not been exposed by any customer until now even though the problem has existed in Rdb since the beginning of the zigzag match feature.

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

1. The main select query joins two tables using a zigzag match strategy with two equality join predicates and one filter predicate.
2. Both the inner and outer keys have the same start and end segments but the difference is that the outer index T2\_NDX has another segment in the middle, ie C2.

```

Outer loop     T1_NDX (C1, C3)
Inner loop     T2_NDX (C1, C2, C3)

```

3. The values of column C1 in both inner and outer loops are ascending. The column C3 of the inner loop contains the same value, but the value of the column C2 in the first row is bigger than that of the third row.
4. The second row in the inner loop is the only row that does not match any rows in the outer loop.

As a workaround, the query returns correctly if the zigzag match is disabled by setting the SQL flag 'nozigzag\_match'.

```

SELECT T2.C1, T2.C2, T2.C3
FROM T2 JOIN T1
ON T1.C1 = T2.C1 AND T1.C3 = T2.C3
WHERE T1.C3 = 1 ;
Tables:
  0 = T2
  1 = T1
Conjunct: (1.C1 = 0.C1) AND (1.C3 = 0.C3)
Match
  Outer loop
    Conjunct: 1.C3 = 1
    Index only retrieval of relation 1:T1
      Index name  T1_NDX [0:0]
  Inner loop
    Conjunct: 0.C3 = 1
    Index only retrieval of relation 0:T2
      Index name  T2_NDX [0:0]
  T2.C1      T2.C2      T2.C3
  10          2          1
  20          1          1
  30          5          1
3 rows selected

```

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

## 3.1.20 Query Slows Down Significantly After Upgrade from Rdb Release 7.0.7.3

Bugs 4597186, 3719771 and 4117082

The following query takes significantly longer to run in Oracle Rdb Release 7.0.8.1 than in Rdb Release 7.0.7.3. The query is a select from a complex view of other views which ends up as being a join of seven tables.

```

select count(*) from A_VIEW
  where RZ = 'U' and TI = 2223581 and UTYPE = 'C';
Tables:
  0 = TBL_T1
  1 = TBL_HLD
  2 = TBL_PFL
  3 = TBL_CPOS
  4 = TBL_EXR
  5 = TBL_PSET
  6 = TBL_TKT
Aggregate: 0:COUNT (*)
Conjunct: (0.RZ = 'U') AND (0.TI = 2223581)
Cross block of 2 entries          (Left Outer Join)
  Cross block entry 1
    Cross block of 2 entries
      Cross block entry 1
        Merge of 1 entries
          Merge block entry 1
            Cross block of 2 entries
              Cross block entry 1
                Leaf#01 BgrOnly 0:TBL_T1 Card=1103
                  Bool: (0.RZ = 'U') AND (0.TI = 2223581)
                    BgrNdx1 TBL_T1_INDEX [2:2] Fan=9
                      Keys: (0.RZ = 'U') AND (0.TI = 2223581)
                        Bool: (0.RZ = 'U') AND (0.TI = 2223581)
              Cross block entry 2
                Index only retrieval of relation 1:TBL_HLD
                  Index name  TBL_HLD_INDEX [4:4]
                    Keys: (0.RZ = 1.RZ) AND (0.TDATE = 1.TDATE) AND
                          (0.BROOT = 1.BROOT) AND (0.BSEC = 1.BSEC)
      Cross block entry 2
        Conjunct: (0.RZ = 'U') AND (0.TI = 2223581)
        Conjunct: (0.RZ = 2.RZ) AND (0.TDATE = 2.TDATE) AND
                  (0.SDATE = 3.SDATE) AND (1.PFL = 2.PFL)
    Cross block of 2 entries          (Left Outer Join)
      Cross block entry 1
        Cross block of 2 entries
          Cross block entry 1
            Reduce: 3.RZ, 4.TDATE, 4.UTYPE, 3.SDATE, 3.PFL
            Sort: 3.RZ(a), 4.TDATE(a), 4.UTYPE(a), 3.SDATE(a), 3.PFL(a)
          Cross block of 2 entries
            Cross block entry 1
              Conjunct: 4.UTYPE = 'C'
              Index only retrieval of relation 4:TBL_EXR
                Index name  TBL_EXR_INDEX [1:1]          <== See Note1
                  Keys: 4.RZ = 'U'
            Cross block entry 2
              Conjunct: 0.SDATE = 3.SDATE
              Conjunct: (3.RZ = 4.RZ) AND (3.CRCY = 4.CRCY)
              Conjunct: 3.RZ = 'U'

```

## Oracle® Rdb for OpenVMS

```
Index only retrieval of relation 3:TBL_CPOS
  Index name  TBL_CPOS_INDEX [2:2]          <== See Note2
  Keys: (3.RZ = 4.RZ) AND (0.SDATE = 3.SDATE)
Cross block entry 2
  Conjunct: 2.STAT <> 5
  Conjunct: (0.RZ = 2.RZ) AND (0.TDATE = 2.TDATE)
  Conjunct: 1.PFL = 2.PFL
  Get      Retrieval by index of relation 2:TBL_PFL
  Index name  TBL_PFL_INDEX [4:4]  Direct lookup
  Keys: (2.RZ = 3.RZ) AND (2.TDATE = 4.TDATE) AND
        (2.UTYPE = 4.UTYPE) AND (2.PFL = 3.PFL)
  Bool: (2.RZ = 'U') AND (2.UTYPE = 'C')
Cross block entry 2
  Conjunct: (2.RZ = 5.RZ) AND (2.PFL = 5.PFL)
  Index only retrieval of relation 5:TBL_PSET
  Index name  TBL_PSET_INDEX [2:2]          Direct lookup
  Keys: (2.RZ = 5.RZ) AND (2.PFL = 5.PFL)
Cross block entry 2
  Conjunct: (0.RZ = 6.RZ) AND (0.TI = 6.TI) AND (1.PFL = 6.PFL)
  Index only retrieval of relation 6:TBL_TKT
  Index name  TBL_TKT_INDEX [3:3]  Direct lookup
  Keys: (0.RZ = 6.RZ) AND (0.TI = 6.TI) AND (1.PFL = 6.PFL)

4489
1 row selected
```

A\_VIEW is defined as follows.

```
CREATE VIEW A_VIEW (RZ, TDATE, UTYPE, SDATE, PFL, TI) AS
(select C1.RZ, C1.TDATE, C4.UTYPE, C1.SDATE, C1.PFL, C1.TI
 from
  (select C2.RZ, C2.TDATE, C2.BROOT, C2.BSEC, C2.SDATE, C2.TI, C3.PFL
   from TBL_T1 as C2 inner join TBL_HLD as C3
     on (C2.RZ = C3.RZ and
         C2.TDATE = C3.TDATE and
         C2.BROOT = C3.BROOT and
         C2.BSEC = C3.BSEC)
   as C1 (RZ, TDATE, BROOT, BSEC, SDATE, TI, PFL)
   inner join
     P_VIEW as C4
     on (C1.RZ = C4.RZ and
         C1.TDATE = C4.TDATE and
         C1.SDATE = C4.SDATE and
         C1.PFL = C4.PFL)
   left outer join
     TBL_TKT as C5
     on (C1.RZ = C5.RZ and
         C1.TI = C5.TI and
         C1.PFL = C5.PFL)
   where (C4.STAT <> 5));
```

```
CREATE VIEW P_VIEW (RZ, TDATE, UTYPE, SDATE, PFL, STAT) AS
(select C1.RZ, C1.TDATE, C1.UTYPE, C2.SDATE, C1.PFL, C1.STAT
 from TBL_PFL as C1
   inner join
     C_VIEW as C2
     on (C1.RZ = C2.RZ and
         C1.TDATE = C2.TDATE and
         C1.UTYPE = C2.UTYPE and
         C1.PFL = C2.PFL)
   left outer join
```

## Oracle® Rdb for OpenVMS

```
PFL_SETUP as C3
on (C1.RZ = C3.RZ and
    C1.PFL = C3.PFL));

CREATE VIEW C_VIEW (RZ, TDATE, UTYPE, SDATE, PFL) AS
(select C2.RZ, C3.TDATE, C3.UTYPE, C2.SDATE, C2.PFL
 from TBL_CPOS as C2
  inner join
    TBL_EXR as C3
  on (C2.RZ = C3.RZ and
      C2.CRCY = C3.CRCY)
 group by C2.RZ, C3.TDATE, C3.UTYPE, C2.SDATE, C2.PFL);
```

Comparing the above output strategy from Rdb Release 7.0.8.1 with the one from Rdb Release 7.0.7.3, it is observed that the following two instances of differences in the index retrievals for the tables TBL\_EXR and TBL\_CPOS are the reasons why the query slows down significantly; the index retrieval is applied with less number of segment keys (i.e. [1:1] as compared to [3:3]).

(1) The following index retrieval is applied with [1:1]:

```
Cross block entry 1
Conjunct: 4.UTYPE = 'C'
Index only retrieval of relation 4:TBL_EXR
Index name  TBL_EXR_INDEX [1:1]          <== See Note1
Keys: 4.RZ = 'U'
```

Note1::

In Rdb Release 7.0.7.3, the following Dynamic leaf retrieval BgrNdx1 is applied with three index key segments [3:3]:

```
Cross block entry 1
Conjunct: 4.RZ = 'U'
Leaf#03 BgrOnly 4:TBL_EXR Card=1580
Bool: 4.UTYPE = 'C'
BgrNdx1 TBL_EXR_INDEX [3:3] Fan=13      <== [3:3]
Keys: (4.RZ = 0.RZ) AND (4.TDATE = 0.TDATE) AND
      (4.UTYPE = 'C')
Bool: 4.RZ = 'U'
```

(2) The following index retrieval is applied with [2:2]:

```
Cross block entry 2
Conjunct: 0.SDATE = 3.SDATE
Conjunct: (3.RZ = 4.RZ) AND (3.CRCY = 4.CRCY)
Conjunct: 3.RZ = 'U'
Index only retrieval of relation 3:TBL_CPOS
Index name  TBL_CPOS_INDEX [2:2]       <== See Note2
Keys: (3.RZ = 4.RZ) AND (0.SDATE = 3.SDATE)
```

Note2::

In Rdb Release 7.0.7.3, the following Dynamic leaf retrieval BgrNdx1 is applied with four index key segments [4:4]:

```
Cross block entry 2
Conjunct: 0.SDATE = 3.SDATE
Conjunct: (3.RZ = 4.RZ) AND (3.CRCY = 4.CRCY)
Leaf#04 BgrOnly 3:TBL_CPOS Card=2825
Bool: 3.RZ = 'U'
BgrNdx1 TBL_CPOS_INDEX [4:4] Fan=10    <== [4:4]
Keys: (3.RZ = 4.RZ) AND (0.SDATE = 3.SDATE) AND
      (3.PFL = 1.PFL) AND (3.CRCY = 4.CRCY)
```

## Oracle® Rdb for OpenVMS

```
Bool: 3.RZ = 'U'
```

As a workaround, the query runs as fast as Rdb Release 7.0.7.3 if the SQL flag 'NOTTRANSITIVITY' is defined, as in the following example.

```
set flags 'NOTTRANSITIVITY';

select count(*) from A_VIEW
  where RZ = 'U' and TI = 2223581 and UTYPE = 'C';
Tables:
  0 = TBL_T1
  1 = TBL_HLD
  2 = TBL_PFL
  3 = TBL_CPOS
  4 = TBL_EXR
  5 = TBL_PSET
  6 = TBL_TKT
Aggregate: 0:COUNT (*)
Conjunct: (0.RZ = 'U') AND (0.TRADE_INDEX = 2223581)
Cross block of 2 entries      (Left Outer Join)
  Cross block entry 1
    Cross block of 2 entries
      Cross block entry 1
        Merge of 1 entries
          Merge block entry 1
            Cross block of 2 entries
              Cross block entry 1
                Leaf#01 BgrOnly 0:TBL_T1 Card=1103
                  Bool: (0.RZ = 'U') AND (0.TI = 2223581)
                  BgrNdx1 TBL_T1_INDEX [2:2] Fan=9
                  Keys: (0.RZ = 'U') AND (0.TI = 2223581)
                Cross block entry 2
                  Index only retrieval of relation 1:TBL_HLD
                    Index name TBL_HLD_INDEX [4:4]
                    Keys: (0.RZ = 1.RZ) AND (0.TDATE = 1.TDATE) AND
                          (0.BROOT = 1.BROOT) AND (0.BSEC = 1.BSEC)
              Cross block entry 2
                Conjunct: (0.RZ = 'U') AND (0.TI = 2223581)
                Conjunct: (0.RZ = 2.RZ) AND (0.TDATE = 2.TDATE) AND
                          (0.SDATE = 3.SDATE) AND (1.PFL = 2.PFL)
            Cross block of 2 entries      (Left Outer Join)
              Cross block entry 1
                Cross block of 2 entries
                  Cross block entry 1
                    Conjunct: 2.UPDATE_TYPE = 'C'
                    Conjunct: 2.STAT <> 5
                    Leaf#02 BgrOnly 2:TBL_PFL Card=4029
                      Bool: (0.RZ = 2.RZ) AND (0.TDATE = 2.TDATE) AND
                            (1.PFL = 2.PFL)
                      BgrNdx1 TBL_PFL_INDEX [4:4] Fan=10
                      Keys: (0.RZ = 2.RZ) AND (0.TDATE = 2.TDATE) AND
                            (2.UTYPE = 'C') AND (1.PFL = 2.PFL)
                  Cross block entry 2
                    Conjunct: (2.RZ = 3.RZ) AND (2.TDATE = 4.TDATE) AND
                              (2.UTYPE = 4.UTYPE)
                    Reduce: 3.RZ, 4.TDATE, 4.UTYPE, 3.SDATE, 3.PFL
                    Sort: 3.RZ(a), 4.TDATE(a), 4.UTYPE(a), 3.SDATE(a), 3.PFL(a)
                Cross block of 2 entries
                  Cross block entry 1
                    Index only retrieval of relation 4:TBL_EXR
                      Index name TBL_EXR_INDEX [0:0]
```

## Oracle® Rdb for OpenVMS

```
Cross block entry 2
  Index only retrieval of relation 3:TBL_CPOS
    Index name  TBL_CPOS_INDEX [4:4]          <== See Note3
  Direct lookup
    Keys: (3.RZ = 4.RZ) AND (0.SDATE = 3.SDATE) AND
          (2.PFL = 3.PFL) AND (3.CURRENCY = 4.CURRENCY)
Cross block entry 2
  Conjunct: (2.RZ = 5.RZ) AND (2.PFL = 5.PFL)
  Index only retrieval of relation 5:PSET
    Index name  TBL_PSET_INDEX [2:2]          Direct lookup
    Keys: (2.RZ = 5.RZ) AND (2.PFL = 5.PFL)
Cross block entry 2
  Conjunct: (0.RZ = 6.RZ) AND (0.TI = 6.TI) AND (1.PFL = 6.PFL)
  Index only retrieval of relation 6:TBL_TKT
    Index name  TBL_TKT_PFL_INDEX [3:3]      Direct lookup
    Keys: (0.RZ = 6.RZ) AND (0.TI = 6.TI) AND (1.PFL = 6.PFL)

4489
1 row selected
```

Note3::

The retrieval by index name TBL\_CPOS\_INDEX in the outer leg now applies "[4:4]" rather than "[2:2]" as in the strategy of the slow version and thus speeds up the query.

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

1. The main query selects from the view "A\_VIEW" with WHERE clause filter predicates. One of the predicates contains the column "RZ" which is transitive between several join operations.
2. A\_VIEW is a view defined as a join query of three tables and another view P\_VIEW where an ON clause of common join booleans is applied for each join. The type of join does not matter much here; it could be either INNER or OUTER join. What seems to matter more here is the presence of common join booleans which form the transitivity selections between several join operations. For example, the following is the transitivity selections between the join booleans from the views A\_VIEW, P\_VIEW and C\_VIEW:

Transitivity selections generated from the views:

```
C1.RZ = C2.RZ = C3.RZ = C4.RZ
C1.TDATE = C2.TDATE = C3.TDATE = C4.TDATE
C1.PFL = C2.PFL = C3.PFL = C4.PFL = C5.PFL
```

Common join booleans from ON clauses of A\_VIEW:

```
on (C2.RZ = C3.RZ and
    C2.TDATE = C3.TDATE and
    C2.BROOT = C3.BROOT and
    C2.BSEC = C3.BSEC)

on (C1.RZ = C4.RZ and
    C1.TDATE = C4.TDATE and
    C1.SDATE = C4.SDATE and
    C1.PFL = C4.PFL)

on (C1.RZ = C5.RZ and
    C1.TI = C5.TI and
    C1.PFL = C5.PFL)
```

Common join booleans from ON clauses of P\_VIEW:

```
on (C1.RZ = C2.RZ and
    C1.TDATE = C2.TDATE and
    C1.UTYPE = C2.UTYPE and
```

## Oracle® Rdb for OpenVMS

```
C1.PFL = C2.PFL)
```

```
on (C1.RZ = C3.RZ and
    C1.PFL = C3.PFL)
```

Common join booleans from ON clauses of C\_VIEW:

```
on (C2.RZ = C3.RZ and
    C2.CRCY = C3.CRCY)
```

3. One of the views (C\_VIEW) contains a GROUP BY clause with the columns being applied in the common join booleans of the ON clause, as in the following example.

```
group by C2.RZ, C3.TDATE, C3.UTYPE, C2.SDATE, C2.PFL);
```

4. The values of the optimizer's statistics and cardinalities of all tables and indices come to play a more important role in this query so that the outcome of the strategy consists of nested Cross (and/or) Match blocks where both the joined tables (TBL\_EXR and TBL\_CPOS) of C\_VIEW with GROUP BY clause and the Left Outer Join of P\_VIEW fall under the inner leg of the topmost Cross joined operation of A\_VIEW.

The following shows the skeleton structure of the strategy with those non-essential parts removed and replaced by the connotation "...etc...".

```
Cross block of 2 entries          (Left Outer Join)  <== represents A_VIEW
Cross block entry 1              <== Inner leg of topmost Cross join
Cross block of 2 entries
Cross block entry 1
...etc...

Cross block of 2 entries
Cross block entry 1
  Leaf#01 BgrOnly 0:TBL_T1 Card=1103
  ...etc...

Cross block entry 2
  Index only retrieval of relation 1:TBL_HLD
  ...etc...

Cross block entry 2
...etc...
Cross block of 2 entries          (Left Outer Join)  <== represents P_VIEW
Cross block entry 1
Cross block of 2 entries
Cross block entry 1              <=== represents C_VIEW
Reduce: 3.RZ, 4.TDATE, 4.UTYPE, 3.SDATE, 3.PFL <== GROUP BY
Sort: 3.RZ(a), 4.TDATE(a), 4.UTYPE(a), 3.SDATE(a), 3.PFL(a)
Cross block of 2 entries
Cross block entry 1
  ...etc...
  Index only retrieval of relation 4:TBL_EXR
  Index name  TBL_EXR_INDEX [1:1]      <== See Note1
  Keys: 4.RZ = 'U'
Cross block entry 2
  ...etc...
  Index only retrieval of relation 3:TBL_CPOS
  Index name  TBL_CPOS_INDEX [2:2]    <== See Note2
  Keys: (3.RZ = 4.RZ) AND (0.SDATE = 3.SDATE)
Cross block entry 2
  ...etc...
  Get      Retrieval by index of relation 2:TBL_PFL
  ...etc...
```

## Oracle® Rdb for OpenVMS

```
Cross block entry 2
  ...etc...
  Index only retrieval of relation 5:TBL_PSET
  ...etc...

Cross block entry 2
  ...etc...
  Index only retrieval of relation 6:TBL_TKT
  ...etc...
```

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

## 3.2 SQL Errors Fixed

### 3.2.1 Unexpected Column Ordering Generated by ALTER TABLE ... ALTER COLUMN Statements

In prior versions of Oracle Rdb, when multiple ALTER COLUMN { BEFORE | AFTER } COLUMN clauses were used in an ALTER TABLE, the resulting column order would be different from that created by multiple ALTER TABLE statements which included just one ALTER COLUMN clause.

The following example shows the problem: both sets of ALTER TABLE statements should result in the same column ordering.

```
SQL> create table PERSON
cont>     (address char(40)
cont>     ,last_name char(30)
cont>     ,social_security_number integer
cont>     ,first_name char(30)
cont>     );
SQL> commit;
SQL>
SQL> show table (column) PERSON;
Information for table PERSON

Columns for table PERSON:
Column Name                Data Type                Domain
-----
ADDRESS                    CHAR(40)
LAST_NAME                  CHAR(30)
SOCIAL_SECURITY_NUMBER     INTEGER
FIRST_NAME                 CHAR(30)

SQL>
SQL> alter table PERSON
cont>     alter column first_name
cont>     before column last_name;
SQL> alter table PERSON
cont>     alter column social_security_number
cont>     after column address;
SQL>
SQL> show table (column) PERSON;
Information for table PERSON
Columns for table PERSON:
Column Name                Data Type                Domain
-----
ADDRESS                    CHAR(40)
SOCIAL_SECURITY_NUMBER     INTEGER
FIRST_NAME                 CHAR(30)
LAST_NAME                  CHAR(30)

SQL>
SQL> rollback;
SQL>
SQL>
SQL> alter table PERSON
cont>     alter column first_name
cont>     before column last_name
cont>     alter column social_security_number
```

```

cont>          after column address;
SQL>
SQL> show table (column) PERSON;
Information for table PERSON

Columns for table PERSON:
Column Name          Data Type          Domain
-----
ADDRESS              CHAR(40)
FIRST_NAME           CHAR(30)
SOCIAL_SECURITY_NUMBER  INTEGER
LAST_NAME            CHAR(30)

SQL>
SQL> rollback;

```

This problem has been corrected in Oracle Rdb Release 7.1.4.2. Oracle Rdb now preserves the relative ordering of columns when multiple ALTER COLUMN clauses are used.

## 3.2.2 Unexpected Errors when Accessing Declared Local Temporary Tables

Bug 4622609

In prior releases of Oracle Rdb, access to the 7th declared local temporary table in a module, Interactive SQL session, or Dynamic SQL session could fail with unexpected errors. Rdb includes an optimization for the RDB\$DATABASE table (which has RDB\$RELATION\_ID of 7) which was incorrectly applied to the 7th declared local temporary table.

The following example shows the error when the column is numeric.

```

declare :x integer;
select avg(c1) into :x from update_db.module.jccdp$_0_1;
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-COSI-F-INVNBDS, invalid numeric byte data string

```

A workaround to this problem is to declare but not use the 7th local temporary table.

This problem has been corrected in Oracle Rdb Release 7.1.4.2. The optimization is now correctly applied only to RDB\$DATABASE and no longer affects declared temporary table usage.

## 3.2.3 SQL\$MOD Bugcheck for Undefined Function Parameter

Bug 4364926

If a SQL stored function defined in a CREATE MODULE statement or an external function defined in a CREATE FUNCTION were referenced in a SQL Module Language procedure or in a precompiled SQL program AND one of the actual parameters was not defined, SQL\$MOD or SQL\$PRE would fail with an access violation and generate a SQLBUGCHK.DMP immediately after reporting the undefined parameter.

The following SQL Module language program shows an example of a coding error which would have resulted in this type of failure:

```
module TEST_BAD_PARAM_REF
  language general
  parameter colons
declare pers alias for filename 'personnel'
declare transaction read only

procedure failure_procedure
  sqlca,
  :pl record f1 char(10), f2 char(10) end record;
begin
  set :pl.f1 = ' ';
  set :pl.f2 = mytrim(:undef);
end;
```

In the above program, the stored function "mytrim" is referenced with an actual parameter of ":undef" which is an undefined variable. When the above program was compiled, SQL\$MOD would fail as follows:

```
VMS> SQL$MOD BUG4364926.SQLMOD
      set :pl.f2 = mytrim(:undef);
                          1
%SQL-F-UNDPARAM, (1) Parameter UNDEF is not declared in procedure
FAILURE_PROCEDURE
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DISK1:[USERNAME]SQLBUGCHK.DMP;
```

The SQLBUGCHK.DMP file had a signature similar to the following:

```
***** Exception at 003CFC00 : ADD_CASE_TERM + 000007E0
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=000000000000009E, PC=00000000003CFC00, PS=0000001B
```

This problem has been corrected so that SQL\$MOD and SQL\$PRE now report the undefined name and continue with the compilation.

As a workaround, correct the syntax error.

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

## 3.2.4 DDL Statements Caused Bugchecks with Various Exceptions

Bug 4616034

In some cases, the DDL commands ALTER DOMAIN and ALTER TABLE ... ALTER COLUMN are required to verify constraints affected by the changes to the column. In the case of ALTER DOMAIN, any changes will be propagated to each column based on the domain in the form of an implicit ALTER TABLE ... ALTER COLUMN.

This action might report an error such as the following:

- Various bugcheck dumps, such as the following:

## Oracle® Rdb for OpenVMS

```
***** Exception at 001EF64C : SQL$SIGNAL_STOP + 000007CC
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=0000000090
```

```
***** Exception at 011DE6A8 : COSI_MEM_GET_VM + 00000398
%SYSTEM-F-ROPRAND, reserved operand fault at PC=0000000011DE6A8, PS=00000009
```

```
***** Exception at 010D65E8 : COSI_MEM_GET_VM_2 + 000002F8
%SYSTEM-F-ROPRAND, reserved operand fault at PC=0000000010D65E8, PS=0000000B
```

```
***** Exception at 010B9FE0 : SOR$END_SORT32 + 000000B0
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=0000000090
```

- **SYSTEM-F-ILLEGAL\_SHADOW** errors might be reported (with bugcheck) on a heavily used Alpha system
- Constraints that included calls to SQL functions would fail with an error similar to:

```
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-RTN_FAIL, routine "(unknown)" failed to compile or execute successfully
-RDB-E-BAD_REQ_HANDLE, invalid request handle
```

- If a constraint had been disabled and it included a reference to a column affected by the ALTER statement, then it would be verified. This is no longer performed as it would cause the ALTER to fail.

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

## 3.3 RMU Errors Fixed

### 3.3.1 RMU/COPY Would Increment Roll-Forward Sequence Number

Bug 2808539

If a database was copied using the RMU/COPY command then backed up and restored, the roll-forward sequence number would be incorrectly incremented. If an RMU/RECOVER command was issued against the restored database and the journal specified by RMU/DUMP/HEADER=JOURNAL was used, then no transactions would be applied. If the journal with the prior sequence number was used for recovery, then transactions would be applied.

The following script demonstrates the problem:

```
$ ! Create a simple db with a single journal.
$
$ SQL$
CREATE DATABASE FILENAME TEST;
CREATE TABLE T1 (C1 INT);
COMMIT;
EXIT
$ RMU/SET AFTER/ENABLE/NOLOG/ADD=(NAME=RDB$JOURNAL,FILE=TEST) TEST
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
$ SQL$
ATTACH 'FILENAME TEST';
INSERT INTO T1 VALUES (1);
1 row inserted
COMMIT;
EXIT;
$
$ ! Determine the current sequence numbers.
$
$ RMU/DUMP/HEADER=JOURNAL/OUTPUT=TEST.TMP TEST
$ SEARCH/NOWINDOW TEST.TMP "sequence number is"
    - Current roll-forward sequence number is 0
    - Current backup sequence number is 0
      Backup sequence number is 0
      Backup sequence number is 0
$
$ ! Copy the db.
$
$ RMU/COPY/NOLOG/ROOT=TEST_COPY TEST
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
$
$ ! Notice that the roll-forward sequence number is 0.
$
$ RMU/DUMP/HEADER=JOURNAL/OUTPUT=TEST.TMP TEST_COPY
$ SEARCH/NOWINDOW TEST.TMP "sequence number is"
    - Current roll-forward sequence number is 0
    - Current backup sequence number is 1
$
$ ! Backup and restore the copied db.
$
$ RMU/BACKUP/NOLOG TEST_COPY TEST_COPY
```

## Oracle® Rdb for OpenVMS

```
$ SQL$
  DROP DATABASE FILENAME TEST_COPY;
  EXIT;
$ RMU/RESTORE/NOLOG/NOCDD TEST_COPY
%RMU-I-AIJRSTAVL, 0 after-image journals available for use
%RMU-I-AIJISOFF, after-image journaling has been disabled
%RMU-W-USERECCOM, Use the RMU Recover command. The journals are not available.
$
$ ! Notice that the roll-forward sequence number is now 1.
$
$ RMU/DUMP/HEADER=JOURNAL/OUTPUT=TEST.TMP TEST_COPY
$ SEARCH/NOWINDOW TEST.TMP "sequence number is"
  - Current roll-forward sequence number is 1
  - Current backup sequence number is 1
$
$ ! Do some work on the original db and then backup the journal.
$
$ SQL$
  ATTACH 'FILENAME TEST';
  INSERT INTO T1 VALUES (1);
1 row inserted
  COMMIT;
  EXIT;
$ RMU/BACKUP/AFTER/NOLOG TEST TEST_AIJ_BACKUP_1
%RMU-W-DATACMIT, unjournalized changes made; database may not be recoverable
$
$ ! Do some more work on the original db.
$
$ SQL$
  ATTACH 'FILENAME TEST';
  INSERT INTO T1 VALUES (1);
1 row inserted
  COMMIT;
  EXIT;
$
$ ! Attempt to apply the journal with sequence number 1 to the copied db.
$ ! No transactions are applied.
$
$ RMU/RECOVER/LOG/NOTRACE/ROOT=TEST_COPY TEST
%RMU-I-LOGRECDB, recovering database file DISK:[DIR]TEST_COPY.RDB;1
%RMU-I-LOGOPNAIJ, opened journal file DISK:[DIR]TEST.AIJ;1 at 26-JUL-2005
19:48:52.47
%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed
%RMU-I-LOGRECOVR, 0 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 2 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCESS, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence number
needed will be 2
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-W-NOTRANAPP, no transactions in this journal were applied
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence number
needed will be 1
%RMU-I-AIJNOENABLED, after-image journaling has not yet been enabled
$
$ ! Attempt to apply the journal with sequence numbers 0 and 1 to the copied
$ ! db. An AIJSEQPRI error is displayed but the transactions get applied.
$
$ RMU/RECOVER/LOG/NOTRACE/ROOT=TEST_COPY TEST_AIJ_BACKUP_1, TEST
%RMU-I-LOGRECDB, recovering database file DISK:[DIR]TEST_COPY.RDB;1
%RMU-I-LOGOPNAIJ, opened journal file DISK:[DIR]TEST_AIJ_BACKUP_1.AIJ;1 at
```

```

26-JUL-2005 19:48:52.52
%RMU-W-AIJSEQPRI, AIJ file sequence number 0 created prior to expected
sequence 1
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-LOGRECOVR, 1 transaction committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 1 transaction ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCEP, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence number
needed will be 1
%RMU-I-LOGOPNAIJ, opened journal file DISK:[DIR]TEST.AIJ;1 at 26-JUL-2005
19:48:52.52
%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed
%RMU-I-LOGRECOVR, 2 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCEP, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence number
needed will be 2
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-LOGSUMMARY, total 3 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
%RMU-I-LOGSUMMARY, total 1 transaction ignored
%RMU-I-AIJSUCCEP, database recovery completed successfully
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence number
needed will be 2
%RMU-I-AIJNOENABLED, after-image journaling has not yet been enabled

```

This problem has been corrected in Oracle Rdb Release 7.1.4.2. The roll-forward sequence number will not be incremented in a database that has been copied, moved, or restored if journaling is disabled in the destination database.

### 3.3.2 RMU /UNLOAD Output File Maximum Record Size

Bug 4573066

Previously, the RMU /UNLOAD command when writing in TEXT format would sometimes incorrectly set the output file maximum record size field in file header.

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

### 3.3.3 RMU Extract Did Not Generate Correct Routine References for Multischema Databases

Bug 4610120

In prior releases of Oracle Rdb, function and procedure references in definitions from a multischema-enabled database were not correctly qualified. This could cause the object definition to fail when re-executed. The following example, edited for brevity, shows the reported problem when a script generated by RMU Extract is executed.

```

SQL> create view CATA.SCHEM.VIEW_M
cont>     stored name is VIEW_M

```

## Oracle® Rdb for OpenVMS

```
cont>      (T_ID,  
...  
cont>      T_NO) as  
cont>      (select  
cont>          C1.T_ID,  
...  
cont>          C1.T_NO  
cont>      from CATA.SCHEM.L_TABLE C1  
cont>      where (C1.T_S_ID = HEX_TO_INT('C84323FE')));  
%SQL-F-SCHNOTDEF, Schema LEE is not defined
```

In this example, the function `HEX_TO_INT` should be fully qualified with catalog and schema names.

This problem has been corrected in Oracle Rdb Release 7.1.4.2. RMU Extract now correctly qualifies the function and procedure references in the generated script.

## **3.4 Row Cache Errors Fixed**

### **3.4.1 Hangs With Row Cache Hash Latches Using Rdb Release 7.1.4.1.1**

Bug 4675955

In Release 7.1.4.1.1 of Oracle Rdb, when using the Row Cache feature, it was possible for applications to hang with a stall reason of "waiting for XXX record cache hash latch". The cause of this problem was tracked down to a code sequence that would not reliably clear the latch bit.

As a workaround, the issue is not seen with the Rdb 7.1.4.1.0 kit. This can be installed.

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

## 3.5 Hot Standby Errors Fixed

### 3.5.1 RDM\$BIND\_HOT\_NETWORK\_RETRY\_COUNT Not Consistently Used

Bug 3000359

The Hot Standby feature will attempt to reconnect to the standby database if the network link is lost. By default it will retry one time. The number of retries can be set to a different number by defining the system-wide logical RDM\$BIND\_HOT\_NETWORK\_RETRY\_COUNT to the desired value. However, depending on the type of network failure, sometimes the defined value would be used, other times it would not.

For example, if the database administrator did not want activity on the master database to stall waiting for a network timeout they might set the number of retries to zero. In prior releases, even with the number of retries set to zero, Hot Standby would still attempt to reconnect to the standby database.

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.4.2. The retry count specified by the RDM\$BIND\_HOT\_NETWORK\_RETRY\_COUNT logical will be obeyed any time that a network link is lost.

### 3.5.2 AIJSIGNATURE Error Not Always Returned When Journals Different

Bug 3422435

The Hot Standby feature requires that the journal configuration on the standby database be identical to the master database. If the configuration is not identical, attempts to start replication would fail with an *%RDMS-F-AIJSIGNATURE* error. For example:

```
%RMU-I-HOTFAILURE, hot standby failure: check number, size of AIJ journals and  
corresponding device type  
%RDMS-F-AIJSIGNATURE, standby database AIJ signature does not match master  
database
```

However, if the journals on the standby were created with allocation sizes that were different from the master, but happened to have the same physical size as the master, no AIJSIGNATURE error was returned. Replication would sometimes start successfully only to shutdown later or a bugcheck would occur.

In previous releases of Oracle Rdb, for fixed size ("circular") journals, when a journal was created the logical end-of-file would be advanced to match physical end-of-file. For example, if a journal was created with an allocation of 513 blocks, and the disk being used had a cluster size of eight blocks, the actual allocated size of the journal would be 520 blocks. Thus the declared allocation size for a journal would often be different from the actual size of the journal.

When Hot Standby replication is started, Oracle Rdb attempts to verify that the journal configurations between the master and the standby database are the same. This is done by comparing the number of journals and the logical end-of-file size for each of the journals. If those characteristics match, then the configuration is considered to be the same. After this validation is done, replication commences and journal entries from the master are copied to the standby database. Hot Standby will attempt to match the journal being copied from the master with the corresponding journal on the standby database by comparing the allocation sizes. However, if no journal on the standby has the same allocation as the journal being replicated from the master, then Hot Standby will fail.

The following example demonstrates the problem. Notice that the journal allocations used on the master database are not the same as the allocations on the standby database. When replication was started, RMU failed with a bugcheck.

```
$ SQL
CREATE DATABASE FILENAME TEST_MASTER
  CREATE STORAGE AREA RDB$SYSTEM FILENAME TEST_MASTER;
DISCONNECT ALL;
ALTER DATABASE FILENAME TEST_MASTER
  RESERVE 2 JOURNALS;
%RDMS-W-DOFULLBCK, full database backup should be done to ensure future recovery
ALTER DATABASE FILE TEST_MASTER JOURNAL IS ENABLED
  (FAST COMMIT IS ENABLED,
   LOG SERVER IS AUTOMATIC)
  ADD JOURNAL AIJ_1 FILENAME TM_AIJ_1 ALLOCATION 513
  ADD JOURNAL AIJ_2 FILENAME TM_AIJ_2 ALLOCATION 514
  ADD JOURNAL AIJ_3 FILENAME TM_AIJ_3 ALLOCATION 515;
%RDMS-W-DOFULLBCK, full database backup should be done to ensure future recovery
EXIT
$ RMU/BACKUP/NOLOG TEST_MASTER TEST_MASTER
$ RMU/RESTORE/NOLOG/NOCCD/NEW/ROOT=TEST_STANDBY -
  /AIJ_OPT=SYS$INPUT TEST_MASTER
  JOURNAL IS ENABLED RESERVE 3
  ADD JOURNAL AIJ_1 FILE TS_AIJ_1 ALLOCATION 516
  ADD JOURNAL AIJ_2 FILE TS_AIJ_2 ALLOCATION 517
  ADD JOURNAL AIJ_3 FILE TS_AIJ_3 ALLOCATION 518
$ DIR /SIZE=ALL *.AIJ
```

Directory DEV:[DIR]

```
TM_AIJ_1.AIJ;1          520/520
TM_AIJ_2.AIJ;1          520/520
TM_AIJ_3.AIJ;1          520/520
TS_AIJ_1.AIJ;1          520/520
TS_AIJ_2.AIJ;1          520/520
TS_AIJ_3.AIJ;1          520/520
```

Total of 6 files, 3120/3120 blocks.

```
$ RMU/OPEN TEST_MASTER
$ RMU/OPEN TEST_STANDBY
$ RMU/REPLICATE AFTER_JOURNAL START TEST_STANDBY -
  /MASTER_ROOT=TEST_MASTER
%RMU-I-HOTOFFLINE, standby database opened for exclusive access
$ RMU/REPLICATE AFTER_JOURNAL START TEST_MASTER -
  /STANDBY_ROOT=TEST_STANDBY
%NONAME-F-NOMSG, Message number 00000004
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-I-BUGCHKDMP, generating bugcheck dump file DEV:[DIR]RMUBUGCHK.DMP;
%RMU-F-FTL_RMU, Fatal error for RMU operation at 19-AUG-2005 14:29:02.81
```

This problem has been resolved. Now, when journals are created, the logical end-of-file is not advanced to the physical end-of-file. That is, the logical end-of-file will now always match the allocation specified when the journal is created. That eliminates the case where the physical journal allocations will match but the declared allocations do not. Attempts to start replication with journals that don't have matching allocations will now properly fail with an *%RDMS-F-AIJSIGNATURE* error.

This problem can be avoided by always using matching allocation sizes when creating the standby journal configuration.

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

### 3.5.3 Standby Journal Disk No Longer Has to Have Same Cluster Size as Master

Bug 3092027

The Hot Standby feature requires that the journal configuration on the standby database be identical to the master database. If the configuration is not identical, attempts to start replication will fail with an *%RDMS-F-AIJSIGNATURE* error. For example:

```
%RMU-I-HOTFAILURE, hot standby failure: check number, size of AIJ journals and  
corresponding device type  
%RDMS-F-AIJSIGNATURE, standby database AIJ signature does not match master  
database
```

In previous releases, to ensure that the journal configurations matched it was necessary to use disks on the standby that had the same cluster size, or a multiple of the cluster size, as the disks used for the master database. In some environments, this could be a difficult requirement to meet.

When Hot Standby replication is started, Oracle Rdb attempts to verify that the journal configurations between the master and the standby database are the same. This is done by comparing the number of journals and the logical end-of-file size for each of the journals. If those characteristics match, then the configuration is considered to be the same.

In previous releases of Oracle Rdb, for fixed size ("circular") journals, when a journal was created the logical end-of-file would be advanced to match physical end-of-file. For example, if a journal was created with an allocation of 513 blocks, and the disk being used had a cluster size of eight blocks, the actual allocated size of the journal would be 520 blocks. Thus the declared allocation size for a journal would often be different from the actual size of the journal.

When the logical end-of-file was being rounded up for new journals, it was necessary to ensure that the disk cluster size on the standby journal disks was compatible with the cluster size on the master journal disks. Failing to do so would often result in *%RDMS-F-AIJSIGNATURE* errors.

This restriction has been lifted. Now, when journals are created, the logical end-of-file is not advanced to the physical end-of-file. That is, the logical end-of-file will now always match the allocation specified when the journal is created. That eliminates the case where the declared allocation sizes of master and standby journals will match but the logical end-of-file sizes do not.

To take advantage of this change, the journals on the master database should be dropped and recreated. The

standby database can then be created using any available disk as long as the journal configuration (number and allocation size of journals) matches the master. Both the master and standby systems must be running the release of Oracle Rdb that has this change.

This problem has been corrected in Oracle Rdb Release 7.1.4.2.

### 3.5.4 Cannot Start Hot Standby if Incremental Restore Done

Bug 4387482

When initially creating a standby database, if an incremental backup is applied to the standby database then attempts to start replication will fail with the following error:

```
%RDMS-F-CANTSTARTLRS, error starting AIJ Log Roll-Forward Server process
-RDMS-F-CANTSTARTLRS, error starting AIJ Log Roll-Forward Server process
```

The Log Recover Server (LRS) would fail because it assumed that the journals currently available on the standby node would contain all changes that are in the current standby database. When it found that the journals did not contain all the transactions, it would fail with the CANTSTARTLRS error.

This problem can be avoided by not applying incremental backups to the standby database.

This problem has been corrected in Oracle Rdb Release 7.1.4.2. The LRS will no longer require that the journals on the standby node contain all transactions. Instead, it will wait until it has synchronized with the master database before checking to ensure that all journal entries needed to rollforward the standby database to its current state are available.

### 3.5.5 RMU/REPLICATE AFTER Qualifiers Ignored

When an RMU/REPLICATE AFTER command was issued without a /STANDBY or /MASTER qualifier, all other qualifiers were accepted but ignored.

For example, in the following command the /ONLINE command would be ignored and the previously configured or default setting would be used:

```
$ RMU/REPLICATE AFTER START /ONLINE STANDBY_DB.RDB
```

If the same command was issued with the /MASTER qualifier, then the /ONLINE qualifier would be obeyed.

```
$ RMU/REPLICATE AFTER START /ONLINE /MASTER=MASTER_DB.RDB STANDBY_DB.RDB
```

This problem has been corrected in Oracle Rdb Release 7.1.4.2. If the database has previously had replication started, or a CONFIGURE command has been previously issued, then all qualifiers will be obeyed, even if /STANDBY and /MASTER are not specified.

---

# **Chapter 4**

## **Software Errors Fixed in Oracle Rdb Release 7.1.4.1**

This chapter describes software errors that are fixed by Oracle Rdb Release 7.1.4.1.

# 4.1 Software Errors Fixed That Apply to All Interfaces

## 4.1.1 Memory Leak With RMU /OPEN /STATISTICS=IMPORT

Starting with Oracle Rdb Release 7.1.2, when using the persistent statistics functionality, the Oracle Rdb monitor process (RDMMON71) would "leak" memory every 30 minutes. The monitor process could eventually run out of P0 virtual address space and could stop accepting new database open requests.

This problem has been corrected in Oracle Rdb Release 7.1.4.1. The monitor process no longer leaks memory when it performs periodic statistics checkpoint operations.

## 4.1.2 Query With Two ORDER BY Clauses Returns Wrong Result

Bugs 4086431, 2882908 and 1329838

The following query, which contains two ORDER BY clauses, returns results in the wrong order.

```
SELECT TMP.LOT_NO, TMP.CID, TMP.TIME FROM
  (SELECT CLOT.LOT_NO,
         CAS.CID,
         T2.TIME
   FROM CLOT AS T1,HLOT AS T2, CAS AS T3,
        (SELECT LOT_NO, MAX(TIME) FROM HLOT GROUP BY LOT_NO)
        AS TMP1 (LOT_NO, TIME)
  WHERE (T1.LOT_STATUS = 'CMP') AND
         T1.LOT_NO = T2.LOT_NO AND
         T1.CID=CAS.CID AND
         T2.LOT_NO = TMP1.LOT_NO AND
         T2.TIME = TMP1.TIME
   ORDER BY T2.LOT_NO) AS TMP
ORDER BY TMP.LOT_NO;
```

Tables:

```
0 = CLOT
1 = HLOT
2 = CAS
3 = HLOT
```

Merge of 1 entries

Merge block entry 1

Cross block of 3 entries

Cross block entry 1

Conjunct: 0.CID = 2.CID

Match

Outer loop (zig-zag)

Index only retrieval of relation 2:CAS

Index name CASI1 [0:0]

Inner loop (zig-zag)

Conjunct: 0.LOT\_STATUS = 'CMP'

Get Retrieval by index of relation 0:CLOT

Index name CLOTI2 [0:0]

Cross block entry 2

## Oracle® Rdb for OpenVMS

```

Index only retrieval of relation 1:HLOT
  Index name  HLOTI1 [1:1]
    Keys: 0.LOT_NO = 1.LOT_NO
Cross block entry 3
  Conjunct: (1.LOT_NO = 3.LOT_NO) AND (1.TIME = <mapped field>)
Merge of 1 entries
  Merge block entry 1
    Aggregate: 0:MAX (3.TIME)
Index only retrieval of relation 3:HLOT
  Index name  HLOTI1 [1:1]
    Keys: 3.LOT_NO = 0.LOT_NO
LOT_NO          CID          TIME
HK1L100152     SH011004    21-DEC-2000 16:23:56.00
HK4L200099     SH011020     1-JUN-2001 11:36:44.00
HK1L100343     SH015004    21-DEC-2000 16:34:21.00
CIMP402051     SK025017     1-DEC-2004 02:45:59.00
MR6NZ6N027     SM091027    18-NOV-2003 10:43:21.00
5 rows selected

```

The query works if the outermost ORDER BY clause is removed.

```

SELECT TMP.LOT_NO, TMP.CID, TMP.TIME FROM
  (SELECT T1.LOT_NO,
    CAS.CID,
    T2.TIME
  FROM CLOT AS T1,HLOT AS T2, CAS AS T3,
    (SELECT LOT_NO, MAX(TIME) FROM HLOT GROUP BY LOT_NO)
    AS TMP1 (LOT_NO, TIME)
  WHERE (T1.LOT_STATUS = 'CMP') AND
    T1.LOT_NO = T2.LOT_NO AND
    T1.CID=CAS.CID AND
    T2.LOT_NO = TMP1.LOT_NO AND
    T2.TIME = TMP1.TIME
  ORDER BY T2.LOT_NO) AS TMP
!ORDER BY TMP.LOT_NO          <== Commented out
;

```

Tables:

```

0 = CLOT
1 = HLOT
2 = CAS
3 = HLOT

```

```

Merge of 1 entries
  Merge block entry 1
Cross block of 4 entries
  Cross block entry 1
    Conjunct: 0.LOT_STATUS = 'CMP'
  Get      Retrieval by index of relation 0:CLOT
    Index name  CLOTI1 [0:0]
Cross block entry 2
  Index only retrieval of relation 2:CAS
    Index name  CASI1 [1:1]      Direct lookup
    Keys: 0.CID = 2.CID
Cross block entry 3
  Index only retrieval of relation 1:HLOT
    Index name  HLOTI1 [1:1]
    Keys: 0.LOT_NO = 1.LOT_NO
Cross block entry 4
  Conjunct: (1.LOT_NO = 3.LOT_NO) AND (1.TIME = <mapped field>)
Merge of 1 entries
  Merge block entry 1
    Aggregate: 0:MAX (3.TIME)
Index only retrieval of relation 3:HLOT

```

```

Index name  HLOTI1 [1:1]
Keys: 3.LOT_NO = 0.LOT_NO
LOT_NO      CID          TIME
CIMP402051  SK025017      1-DEC-2004 02:45:59.00
HK1L100152  SH011004      21-DEC-2000 16:23:56.00
HK1L100343  SH015004      21-DEC-2000 16:34:21.00
HK4L200099  SH011020      1-JUN-2001 11:36:44.00
MR6NZ6N027  SM091027      18-NOV-2003 10:43:21.00
5 rows selected

```

Notice that the problem query applies the index CASI1 with the column 2.CID at the outermost cross block while the good query applies the index CLOTI1 with the column 0.LOT\_NO, which is the correct one referenced by the ORDER BY clause.

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

1. The cursor query contains the outer SELECT statement as a derived table wrapped around the inner SELECT statement joining three tables and an aggregate query with MAX and GROUP BY clause.
2. The inner SELECT statement contains a WHERE clause with four equality predicates and a filter predicate, followed by an ORDER BY clause, referencing the same column as the GROUP BY clause.
3. The outer SELECT statement contains an ORDER BY clause referencing the same column as the inner ORDER BY clause via the derived table.

As a workaround, the query works if the outermost ORDER BY clause is removed, as shown in the example above.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

### 4.1.3 Database Shutdown Interrupted by Re-attach May Cause Lost ALS Process

Bug 2668892

When the database open mode and the ALS are AUTOMATIC, if a new process attaches to the database at about the same time as the last process is detaching and closing the database, then it is possible, in some rare cases, that the ALS will be stopped by the detaching process and not restarted by the attaching process.

As a workaround, you can start the ALS manually when the database is open and the ALS is missing with the following command:

```
$ RMU/SERVER AFTER_JOURNAL START <root-file-spec>
```

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

### 4.1.4 Query With BETWEEN Clause Slows Down Using Index Full Scan

Bug 3835253

## Oracle® Rdb for OpenVMS

A customer found that the following query runs much slower as compared to the previous version of Rdb.

```
SELECT COUNT(*)
FROM T1 V, T2 L, T3 P, T4 A
WHERE
  V.LID = L.LID AND
  L.PID = P.PID AND
  P.AID = A.AID AND
  PSTAT = 'FF' AND
  PNUM = '123456' AND
  V.TX_DATE BETWEEN DATE ANSI '2003-01-01' AND DATE ANSI '2003-01-05' AND
  A.FLAG = 'T';
```

Tables:

```
0 = T1
1 = T2
2 = T3
3 = T4
```

Aggregate: 0:COUNT (\*)

Conjunct: 0.LID = 1.LID

Match

Outer loop

Sort: 1.LID(a)

Cross block of 3 entries

Cross block entry 1

Leaf#01 BgrOnly 3:T4 Card=21

Bool: 3.FLAG = 'T'

BgrNdx1 T4\_NDX [0:0] Fan=21

Cross block entry 2

Conjunct: 2.AID = 3.AID

Get Retrieval sequentially of relation 2:T3

Cross block entry 3

Conjunct: 1.PID = 2.PID

Index only retrieval of relation 1:T2

Index name T2\_nDX [0:0]

Inner loop (zig-zag)

Conjunct: 0.PSTAT = 'FF'

Conjunct: 0.PNUM = '123456'

Conjunct: 0.TX\_DATE >= DATE '2003-01-01'

Conjunct: 0.TX\_DATE <= DATE '2003-01-05'

Get Retrieval by index of relation 0:T1

Index name I\_VIOL\_TX\_SD2 [0:0]

Bool: (0.TX\_DATE >= DATE '2003-01-01') AND (0.TX\_DATE <= DATE  
'2003-01-05')

0

1 row selected

This problem started showing up in Oracle Rdb Release 7.1.2.1 where the fix for Bug 3144382 was included. For Rdb 7.0, the problem started in Rdb Release 7.0.7.1.

This type of performance problem could likely occur in a query with a predicate of either GTR, GEQ, LSS, LEQ, NEQ, CONTAINING or STARTS operator as the leading segment of the index retrieval.

As a workaround, the query correctly applies [2:2] index retrieval if the SQL flag 'selectivity' is set.

```
set flags 'selectivity';
```

```
! run the same query from above
```

```
!
```

Tables:

```

0 = T1
1 = T2
2 = T3
3 = T4
Aggregate: 0:COUNT (*)
Cross block of 4 entries
Cross block entry 1
  Leaf#01 BgrOnly 3:T4 Card=21
    Bool: 3.FLAG = 'T'
    BgrNdx1 T4_NDX [0:0] Fan=21
Cross block entry 2
  Conjunct: 2.AID = 3.AID
  Get      Retrieval sequentially of relation 2:T3
Cross block entry 3
  Conjunct: 1.PID = 2.PID
  Index only retrieval of relation 1:T2
  Index name T2_nDX [0:0]
Cross block entry 4
  Leaf#02 BgrOnly 0:T1 Card=19098133
    Bool: (0.LID = 1.LID) AND (0.PSTAT = 'FF') AND (
          0.PNUM = '123456') AND (0.TX_DATE >= DATE '2003-01-01') AND
          (0.TX_DATE <= DATE '2003-01-05')
    BgrNdx1 I_VIOL_TX_SD2 [2:2] Fan=10
    Keys: (0.LID = 1.LID) AND (0.TX_DATE >= DATE '2003-01-01') AND (
          0.TX_DATE <= DATE '2003-01-05')

0
1 row selected

```

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.5 Database Recovery Process May Fail When AIJs are Full

Bug 4122574

When AIJs were full, it was possible for the database recovery process to fail when trying to recover processes with active read write transactions. The failure occurred when trying to undo the current uncommitted transaction. The DBR log file would show the following:

```

<timestamp> - Starting transaction UNDO for TSN 0:128
<timestamp> - UNDO TSN 0:128 starts at RUJ JFA (2:0)
<timestamp> - Scanning AIJ for optimistic commit
<timestamp> - Starting AIJ scan at 1:513
%RDMS-I-BUGCHKDMP, generating bugcheck dump file SYS$SYSROOT:[SYSEXE]
RDMDBRBUG.DMP

```

Note that the AIJ scan started at 1:513 while the AIJ file is only 512 blocks.

The exception in the dump is:

```

***** Exception at 001B3418 : UTIO$READ_BLOCK + 00000208
%RDMS-F-FILACCERR, error reading disk file
-SYSTEM-W-ENDOFFILE, end of file

```

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.6 Count Distinct(fld) May Fail When Sorted Ranked Indexes are Used

Bug 4160534

In Oracle Rdb Release 7.1.0, an optimization enhancement was added to Rdb to handle "count distinct" type queries if a ranked index could be used to scan the distinct values. This optimization may return incorrect results if the field that the distinct value is based on is a leading segment in a multisegment sorted ranked index.

Count scan can only be used to determine distinct counts if the value expression of count distinct is covered by the entire key of the index, not just the leading segments.

A workaround is to disable COUNT SCAN optimization using the SET Flag statement:

```
SET FLAGS 'NOCOUNT_SCAN'
```

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.7 Left Outer Join View Query With Constant Columns Returns Wrong Result

Bugs 1752645 and 4155086

A customer gets the wrong result with the following query which selects from a left outer join view with constant columns.

```
create view test_vw1 as
select
cast(cast('ABCD' as char(4)) as char(4)) as fld_1,
cast('ABCD' as char(5)) as fld_2,
cast(case when 1=1 then 'YYY' else 'NNN' end as char(3)) as fld_3,
cast('NNN' as char(3)) as fld_4
  from rdb$database t1
 left outer join rdb$database t2 on t2.rdb$file_name = 'asdfasdf'
;
sel * from test_vw1;
Tables:
  0 = RDB$DATABASE
  1 = RDB$DATABASE
Cross block of 2 entries          (Left Outer Join)
Cross block entry 1
  Retrieval sequentially of relation 0:RDB$DATABASE
Cross block entry 2
  Conjunct: 1.RDB$FILE_NAME = 'asdfasdf'
  Get      Retrieval sequentially of relation 1:RDB$DATABASE
FLD_1  FLD_2  FLD_3  FLD_4
NULL   NULL   YYY    NULL
1 row selected
```

The result should display the constant columns instead of NULL. Without the view, the query returns the correct result.

## Oracle® Rdb for OpenVMS

```
select
cast(cast('ABCD' as char(4)) as char(4)) as fld_1,
cast('ABCD' as char(5)) as fld_2,
cast(case when 1=1 then 'YYY' else 'NNN' end as char(3)) as fld_3,
cast('NNN' as char(3)) as fld_4
  from rdb$database t1
  left outer join rdb$database t2 on t2.rdb$file_name = 'asdfasdf'
;
Tables:
  0 = RDB$DATABASE
  1 = RDB$DATABASE
Cross block of 2 entries          (Left Outer Join)
Cross block entry 1
  Retrieval sequentially of relation 0:RDB$DATABASE
Cross block entry 2
  Conjunct: 1.RDB$FILE_NAME = 'asdfasdf'
  Get      Retrieval sequentially of relation 1:RDB$DATABASE
FLD_1   FLD_2   FLD_3   FLD_4
ABCD    ABCD    YYY     NNN
1 row selected
```

The original design for Bug 1752645, which allows a constant column in a view definition for an outer join, caused all types of problems in the optimizer because it could not associate the constant column to the view. With this fix, the wrong result is produced by the simple query (like the one above) which selects constant columns of nested expressions from a view query with left outer join on two tables.

The fix for Bug 1752645 has been withdrawn from all Rdb releases to fix this current problem.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

### 4.1.8 Bugchecks at DIOMARK\$NEW\_SNAP\_PAGE + 000000D0 When Area Added Online

Bug 3818408

If a database was currently open on multiple nodes, and a storage area was added on one node, attempts to reference that area on other nodes could result in a bugcheck containing an exception similar to the following:

```
***** Exception at 011F4500 : DIOMARK$NEW_SNAP_PAGE + 000000D0
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=
0000000000000008, PC=00000000011F4500, PS=0000000B
```

This problem can be demonstrated with the following commands.

Session on Node 1:

```
$ SQL$
SQL> CREATE DATABASE FILENAME TEST OPEN IS MANUAL
SQL>   RESERVE 10 STORAGE AREAS
SQL>
SQL>   CREATE STORAGE AREA RDB$SYSTEM FILENAME TEST;
SQL> EXIT
```

Session on Node 2:

```
$ RMU/OPEN/WAIT TEST
```

Session on Node 1:

```
$ RMU/OPEN/WAIT TEST
$
$ SQL$
SQL> ALTER DATABASE FILENAME TEST
SQL>   ADD STORAGE AREA AREA1 FILENAME AREA1;
SQL>
SQL> ATTACH 'FILENAME TEST';
SQL> CREATE TABLE T1 (F1 INTEGER);
SQL> CREATE STORAGE MAP M1 FOR T1 STORE IN AREA1;
SQL> COMMIT;
SQL> EXIT
```

Session on Node 2:

```
SQL> ATTACH 'FILENAME TEST';
SQL> INSERT INTO T1 VALUES (1);
%RDMS-I-BUGCHKDMP, generating bugcheck dump file dev:[dir]RDSEBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file dev:[dir]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=
0000000000000008, PC=000000000027F20C, PS=0000001B
SQL> ROLLBACK;
SQL> EXIT;
```

The bugcheck would only occur if a specific sequence of actions occurred the first time the new storage area was accessed on a node that did not create the area. Under certain conditions, the data structure describing the new storage area was not being read from disk prior to being accessed.

The problem can be avoided by closing the database on all nodes prior to adding a new storage area.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.9 Query with GROUP BY and ORDER BY Returned Rows in the Wrong Order

Bug 4239808

The following query with GROUP BY and ORDER BY clauses returned rows in the wrong order. The detailed query strategy appears after the select statement and before the resulting data rows. If you compare the order of the keys in the ORDER BY clause of the SELECT statement with the order of those same keys in the Sort: line of the detailed query strategy, you will see that they are not the same. This pinpoints the cause of the error.

```
set flags 'detail,strategy';
select
  T2.ITYPE,
  T3.UNLY,
  SUM(T1.AMT) as AMOUNT
from
  (select CLRH, SCTRY, SMRKT,
```

```

        SIG, SCMD,
        ETYPE, CNO, DSNAME, AMT
    from EDS
    where BID = 1 and
        ETYPE = 1 and
        CNO IN (24,25) ) as T1
join
    EINTR as T2 on
    T2.SCTRY = T1.SCTRY AND
    T2.SMRKT = T1.SMRKT AND
    T2.SIG = T1.SIG
join
    ENT_UND as T3 on
    T3.SCMD = T1.SCMD
group by
    T1.CLRH,
    T2.ITYPE,
    T3.UNLY,
    T1.ETYPE,
    T1.CNO,
    T1.DSNAME
order by
    T1.CLRH,
    T1.DSNAME,
    T1.ETYPE,
    T1.CNO,
    T2.ITYPE,
    T3.UNLY;
T2.ITYPE    T3.UNLY                AMOUNT
HSIC        HSI                    6.660000000000000E+002
HSIP        HSI                    7.140000000000000E+002
SFU2        JSE                    5.684000000000002E+002
MHIC        MHI                    2.000000000000000E+002
MHIF        MHI                    1.840000000000000E+001
MHIP        MHI                    4.000000000000000E+001

    ... etc. ...

12 rows selected

```

The results appear in the correct order when the query is run under Rdb Release 7.0.6.2. The value SFU2 in column T2.ITYPE would correctly appear in row 6 (as shown in the example that follows). In the example above, that value is sorted incorrectly and appears in row 3.

As a workaround, the query works if the columns of the GROUP BY clause are rearranged to be in the same order as in the ORDER BY clause.

```

select
    T2.ITYPE,
    T3.UNLY,
    SUM(T1.AMT) as AMOUNT
from
    (select CLRH, SCTRY, SMRKT,
        SIG, SCMD,
        ETYPE, CNO, DSNAME, AMT
    from EDS
    where BID = 1 and
        ETYPE = 1 and
        CNO IN (24,25) ) as T1
join

```

```

EINTR as INST on
T2.SCTRY = T1.SCTRY AND
T2.SMRKT = T1.SMRKT AND
T2.SIG = T1.SIG
join
ENT_UND as UND on
T3.SCMD = T1.SCMD
group by
T1.CLRH,
T1.DSNAME,
T1.ETYPE,
T1.CNO,
T2.ITYPE,
T3.UNLY
order by
T1.CLRH,
T1.DSNAME,
T1.ETYPE,
T1.CNO,
T2.ITYPE,
T3.UNLY;
T2.ITYPE  T3.UNLY          AMOUNT
HSIC      HSI              6.660000000000000E+002
HSIP      HSI              7.140000000000000E+002
MHIC      MHI              2.000000000000000E+002
MHIF      MHI              1.840000000000001E+001
MHIP      MHI              4.000000000000000E+001
SFU2      JSE              5.684000000000001E+002

... etc. ...

```

12 rows selected

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

1. The select query contains GROUP BY and ORDER BY clauses where the columns of the ORDER BY clause have a different order from that of the GROUP BY clause.
2. One item of the select list is an aggregate of some sort, for example SUM.
3. The query applies the cross strategy rather than a match strategy.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.10 Query with GROUP BY, ORDER BY Returned Rows in the Wrong Order

Bugs 4239808 and 3197004

A query with a GROUP BY and an ORDER BY clause presented its rows sorted in the wrong order. Below is an example query on the standard Rdb PERSONNEL database. The example first shows the SQL query, followed by a detailed diagnostic output of the Rdb optimizer's query strategy, followed by the rows returned as the result of the query. Certain lines in the example have been annotated with numerals towards the right margin so that reference can be made to those lines in the explanation that follows the example.

```

set flags 'detail,strategy';

select employee_id as emp_id,

```

```

        manager_id as mgr_id,
        last_name as last_name
    from employees inner join departments
    on (employee_id = manager_id)
    group by employee_id, last_name, manager_id
    order by employee_id desc, manager_id desc;
1
2
3
Tables:
    0 = EMPLOYEES
    1 = DEPARTMENTS
Reduce: 0.LAST_NAME, 0.EMPLOYEE_ID, 1.MANAGER_ID
Sort: 0.LAST_NAME(a), 0.EMPLOYEE_ID(d), 1.MANAGER_ID(a)
4
Cross block of 2 entries
    Cross block entry 1
        Get Retrieval sequentially of relation 1:DEPARTMENTS
    Cross block entry 2
        Get Retrieval by index of relation 0:EMPLOYEES
        Index name  EMP_EMPLOYEE_ID [1:1] Direct lookup
        Keys: 0.EMPLOYEE_ID = 1.MANAGER_ID
EMP_ID  MGR_ID  LAST_NAME
00374   00374   Andriola
00207   00207   Babbin
00205   00205   Bartlett
00173   00173   Bartlett

...

26 rows selected

```

The query strategy, in this case, showed but a single Sort: line (the line marked 4). The order of the sort keys was different from the order in the query's ORDER BY clause (line 3). The query contains a GROUP BY clause (line 2), which implies that sorting must be done to perform the grouping operation. As a rule, the Rdb optimizer tries to rearrange keys in a GROUP BY sort using the order of the keys in an outer ORDER BY sort. If it can do so, Rdb eliminates one of the two sorts by combining them. In this example, Rdb did so incorrectly, resulting in the wrong order of sort keys in the remaining sort operation of the original two. This explains why the rows were sorted incorrectly.

In the examples, the EMPLOYEE\_ID and MANAGER\_ID columns are used in an ON clause (line 1) as the condition for joining two tables, EMPLOYEES and DEPARTMENTS. The EMPLOYEE\_ID and MANAGER\_ID columns appear in the ORDER BY clause (line 3) and those sort keys are considered to be equivalent. One can sort on the one key or sort on the other to achieve the same results. In certain cases (the preceding example showing one of them), the fact that the ORDER BY clause contained two or more equivalent sort keys was one of the conditions that caused the incorrect behavior in Rdb.

The following example shows the corrected behavior.

```

set flags 'detail,strategy';

select employee_id as emp_id,
        manager_id as mgr_id,
        last_name as last_name
    from employees inner join departments
    on (employee_id = manager_id)
    group by employee_id, last_name, manager_id
    order by employee_id desc, manager_id desc;
1
2
3
Tables:
    0 = EMPLOYEES
    1 = DEPARTMENTS

```

```

Sort: 0.EMPLOYEE_ID(d), 1.MANAGER_ID(d) 5
Reduce: 0.EMPLOYEE_ID, 0.LAST_NAME, 1.MANAGER_ID
Sort: 0.EMPLOYEE_ID(a), 0.LAST_NAME(a), 1.MANAGER_ID(a) 4
Cross block of 2 entries
  Cross block entry 1
    Get Retrieval sequentially of relation 1:DEPARTMENTS
  Cross block entry 2
    Get Retrieval by index of relation 0:EMPLOYEES
      Index name EMP_EMPLOYEE_ID [1:1] Direct lookup
      Keys: 0.EMPLOYEE_ID = 1.MANAGER_ID
EMP_ID  MGR_ID  LAST_NAME
00471   00471   Herbener
00418   00418   Blount
00405   00405   Dement
00374   00374   Andriola

...

26 rows selected

```

This second example, above, shows the query results returned in the desired order. The query strategy now has two sorts performed (one at line 4 for the GROUP BY clause and one at line 5 for the ORDER BY clause).

A third example, below, shows a variation of the previous query. In this next example, the columns in the GROUP BY and the ORDER BY clauses are the same, and the order sorts first in descending order on the first key and in ascending order on the second key. See the line marked 1 in the right margin for the sort order that Rdb used. Sorting first on MANAGER\_ID and then on EMPLOYEE\_ID is acceptable because the two columns are equivalent as explained earlier in this note. The problem was that the sorting was done in ascending order.

```

set flags 'detail,strategy';

select employee_id as emp_id,
       manager_id as mgr_id
  from employees inner join departments
    on (employee_id = manager_id)
  group by employee_id, manager_id
  order by employee_id desc, manager_id asc;
Tables:
  0 = EMPLOYEES
  1 = DEPARTMENTS
Reduce: 1.MANAGER_ID, 0.EMPLOYEE_ID
Sort: 1.MANAGER_ID(a), 0.EMPLOYEE_ID(a) 1
Cross block of 2 entries
  Cross block entry 1
    Get Retrieval sequentially of relation 1:DEPARTMENTS
  Cross block entry 2
    Index only retrieval of relation 0:EMPLOYEES
      Index name EMP_EMPLOYEE_ID [1:1] Direct lookup
      Keys: 0.EMPLOYEE_ID = 1.MANAGER_ID
EMP_ID  MGR_ID
00164   00164
00166   00166
00168   00168
00173   00173

...

26 rows selected

```

The following example shows the corrected behavior. Again, the positions of `MANAGER_ID` and `EMPLOYEE_ID` are swapped in the sort order, which is permissible given that the two columns are equivalent in this query. The first key is sorted in descending order and the second key is sorted in ascending order, as they should be.

```

set flags 'detail,strategy';

select employee_id as emp_id,
       manager_id as mgr_id
  from employees inner join departments
    on (employee_id = manager_id)
  group by employee_id, manager_id
  order by employee_id desc, manager_id asc;
Tables:
  0 = EMPLOYEES
  1 = DEPARTMENTS
Reduce: 1.MANAGER_ID, 0.EMPLOYEE_ID
Sort: 1.MANAGER_ID(d), 0.EMPLOYEE_ID(a)
Cross block of 2 entries
  Cross block entry 1
    Get Retrieval sequentially of relation 1:DEPARTMENTS
  Cross block entry 2
    Index only retrieval of relation 0:EMPLOYEES
    Index name EMP_EMPLOYEE_ID [1:1] Direct lookup
    Keys: 0.EMPLOYEE_ID = 1.MANAGER_ID
EMP_ID  MGR_ID
00471   00471
00418   00418
00405   00405
00374   00374
...
26 rows selected

```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.11 Truncating Empty Table Leaves Uncommitted Transaction in Journal

Bug 4245771

If an empty table was truncated, an after-image journal (AIJ) entry would be made for the truncate operation but no commit entry would be entered in the journal for the committed transaction. Thus the transaction would appear uncommitted when a recover operation was done with the journal.

For example, the following message could be displayed by a `RMU/RECOVER` command when recovering a journal that has missing `COMMIT` entries:

```
%RMU-I-LOGRECSTAT, transaction with TSN nn:nn is active
```

This message can be safely ignored. To prevent this from occurring, some other kind of update operation would need to be done in the same transaction as the `TRUNCATE` command. For example, if a row was

added to the table prior to the TRUNCATE command then that would prevent this problem from occurring.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.12 Bugchecks in AIJUTL\$FREE\_DIRTY\_ARBS When Journals Full

Bug 4088221

Various processes could fail with bugchecks in AIJUTL\$FREE\_DIRTY\_ARBS if all journals became full, a user process was terminated, a journal was backed up, and further journaling activity in the database occurred.

The bugcheck exception was similar to the following:

```
***** Exception at 011E9E94 : AIJUTL$FREE_DIRTY_ARBS + 000005F4
%COSI-F-BUGCHECK, internal consistency failure
```

This problem was introduced in Oracle Rdb Release 7.1.2.4.

The following demonstrates how this problem could occur.

Create a simple database with two minimum sized journals:

```
$ SQL$
CREATE DATABASE FILENAME TEST
  CREATE STORAGE AREA AREA1
    FILENAME AREA1 ;
CREATE TABLE TABLE1 (COLUMN1 INTEGER, COLUMN2 CHAR (900));
CREATE STORAGE MAP MAP1 FOR TABLE1
  DISABLE COMPRESSION
  STORE IN AREA1;
COMMIT;
DISCONNECT ALL;

ALTER DATABASE FILE TEST
  JOURNAL IS ENABLED
  (FAST COMMIT IS ENABLED,
  SHUTDOWN TIME IS 1 MINUTE)
  RESERVE 1 JOURNAL
  ADD JOURNAL JOURNAL1 FILENAME 'JOURNAL1'
  ADD JOURNAL JOURNAL2 FILENAME 'JOURNAL2';
EXIT;
```

Fill the journals:

```
$ RMU/OPEN TEST
$ SQL$
ATTACH 'FILENAME TEST';

INSERT INTO TABLE1 VALUES (1, 'Text');
COMMIT;
EXIT;
$ RMU/SET AFTER_JOURNAL /SWITCH TEST
$ SQL$
ATTACH 'FILENAME TEST$DATABASE:TEST';
BEGIN
```

```

DECLARE :I INT;
FOR :I IN 2 TO 100000
DO
  INSERT INTO TABLE1 VALUES (:I, 'Text');
END FOR;
COMMIT;
END;

```

After the journals fill, the insert process will hang. Kill it, backup the journals, and then attempt to insert more data into the table:

```

$ RMU/BACKUP/AFTER/NOLOG TEST NL:
$ SQL$
ATTACH 'FILENAME TEST';

INSERT INTO TABLE1 VALUES (1, 'Text');
%RDMS-I-BUGCHKDMP, generating bugcheck dump file dev:[dir]RDSBUGCHK.DMP;
%COSI-F-BUGCHECK, internal consistency failure

```

This problem can be avoided by ensuring that journals are backed up prior to all journals becoming full.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.13 Query With Shared Expressions in OR Predicates Returns Wrong Result

Bugs 4300529 and 3918278

The following query with shared expressions in the OR predicate returns the wrong result.

```

select * from
  (SELECT T1.SEM, T1.PAL, T1.VUO, T1.KK, T1.PV
   FROM T1, T2
   WHERE (T2.SEM = T1.SEM)
        AND
        ((NOT EXISTS (SELECT T3.SEM FROM T3
                      WHERE T3.SEM = T1.SEM))
         OR
         (T1.SEM = 'JOVK'))
        AND
        ((NOT EXISTS (SELECT T3.SEM FROM T3
                      WHERE T3.PAL = T1.PAL))
         OR
         (T1.SEM = 'JOVK'))
   ) as DTAB (SEM, PAL, VUO, KK, PV)
where VUO = '2005' and KK = '03' and PV = '29';
Tables:
  0 = T1
  1 = T2
  2 = T3
  3 = T3
Merge of 1 entries
Merge block entry 1
Cross block of 4 entries
Cross block entry 1
  Conjunct: (0.VUO = '2005') AND (0.KK = '03') AND (0.PV = '29')
  Index only retrieval of relation 0:T1

```

## Oracle® Rdb for OpenVMS

```
Index name T1_INDEX [3:3]
Keys: (0.VUO = '2005') AND (0.KK = '03') AND (0.Pv = '29')
Cross block entry 2
Conjunct: (<agg0> = 0) OR (0.SEM = 'JOVK')
Conjunct: (<agg0> = 0) OR (0.SEM = 'JOVK')
Aggregate-F1: 0:COUNT-ANY (<subselect>)
Conjunct: (2.SEM = 0.SEM)
Index only retrieval of relation 2:T3
Index name T3_INDEX [0:0]
Cross block entry 3
Conjunct: (<agg0> = 0) OR (0.SEM = 'JOVK')
Conjunct: 0.SEM = 'JOVK' <== See NOTE
Conjunct: ((<agg0> = 0) OR (0.SEM = 'JOVK')) AND
((<agg1> = 0) OR (0.SEM = 'JOVK'))
Aggregate-F1: 1:COUNT-ANY (<subselect>)
Conjunct: (<agg0> = 0) OR (0.SEM = 'JOVK')
Conjunct: (<agg0> = 0) OR (0.SEM = 'JOVK')
Conjunct: (3.PAL = 0.PAL)
Index only retrieval of relation 3:T3
Index name T3_INDEX [0:0]
Cross block entry 4
Conjunct: ((<agg0> = 0) OR (0.SEM = 'JOVK')) AND
((<agg1> = 0) OR (0.SEM = 'JOVK'))
Index only retrieval of relation 1:T2
Index name T2_INDEX [1:1] Direct lookup
Keys: 1.SEM = 0.SEM
SEM PAL VUO KK PV
JOVK TV2 2005 03 29
JOVK TV2 2005 03 29
2 rows selected
```

NOTE:: The conjunct "(0.SEM = 'JOVK')" is separated from its parent OR predicate with the other left operand "<agg0> = 0)".

The following cross entry 3 contains the incorrect conjunct "(0.SEM = 'JOVK')" which is separated from the other left operand "<agg0> = 0)" of the OR predicate.

```
Cross block entry 3
Conjunct: (<agg0> = 0) OR (0.SEM = 'JOVK')
Conjunct: 0.SEM = 'JOVK' <== Incorrect
Conjunct: ((<agg0> = 0) OR (0.SEM = 'JOVK')) AND
((<agg1> = 0) OR (0.SEM = 'JOVK'))
Aggregate-F1: 1:COUNT-ANY (<subselect>)
Conjunct: (<agg0> = 0) OR (0.SEM = 'JOVK')
Conjunct: (<agg0> = 0) OR (0.SEM = 'JOVK')
Conjunct: (3.PAL = 0.PAL)
Index only retrieval of relation 3:T3
Index name T3_INDEX [0:0]
```

There is no known workaround for this problem.

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

1. The main query selects from a derived table joining two tables with the filtering predicates in the outer WHERE clause.
2. The inner WHERE clause of the query selecting the derived table contains a join equality predicate and two similar OR predicates with a shared expression "(T1.SEM = 'JOVK')" as the right side operand.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.14 Failed Users Not Recovered if DBR Startup Fails

Failed users would not always be recovered if a database shutdown was forced due to an error when the database monitor attempted to create a database recovery (DBR) process.

For example, if there were insufficient process slots available on the system to create another process and the database monitor could not create a DBR then the database would be shutdown. The forced shutdown would leave behind user entries in the database for all the processes that were accessing the database from that node. However, the monitor would incorrectly clear the entry in the data structure used to determine if a node recovery was needed. This would prevent the failed users from being recovered the next time that the database was accessed.

Since the failed users were not recovered, it is possible that database corruption could be introduced since changes made by the failed users were not rolled back before other users accessed the data. It is possible that logical inconsistencies could have been introduced in the database that cannot be detected by RMU/VERIFY. If this problem is encountered, it is advised that the database be restored from the last backup and after-image journals be applied to recover the database up to the point of failure. Recovery past the point of failure may be possible but could re-introduce corruption.

This problem can be avoided by ensuring that there are sufficient system resources available to start DBR processes when needed.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.15 Various Errors or Corruption of Ranked Indexes

Bug 4216643

If a series of updates occurred on an index node of *type is sorted ranked* in the same transaction, it was possible that the index node could be corrupted.

Several different results could occur depending on subsequent operations in the transaction.

- If there were no further operations on the index node in the same transaction and the transaction committed, the index will be left corrupt.
- If a subsequent update attempted certain operations on the index node, various bugchecks could result. In this case, the transaction would be rolled back and the index node would not be corrupt.
- If a subsequent update attempted to update the same index node, the update could fail with the error message "RDB-E-NO\_RECORD, access by dbkey failed because dbkey is no longer associated with a record". In this case, the index may or may not be left corrupt depending on the precise sequence of operations and also on the application's response to the error. If the application commits the transaction, the index may be left corrupt. If the application rolls back the transaction, the index will not be left corrupt.

In the reported case, a RDB-E-NO\_RECORD error was reported and the failed update was automatically rolled back. This is termed a *verb rollback*. The following example shows the reported error.

```
SQL> delete from some_table where some_key < 20050225;
```

```
%RDB-E-NO_RECORD, access by dbkey failed because dbkey is no longer
associated with a record
-RDMS-F-NODBK, 98:770:1 does not point to a data record
```

In this case, a subsequent RMU/VERIFY reported no errors and the index was not corrupt.

To produce this problem, a precise sequence of operations needed to occur within the same transaction.

- A row is deleted where the key value is not the first key value in the index node and that key value has many duplicates causing the duplicates chain to overflow into several overflow index nodes.
- The dbkey for the deleted row must be in the first overflow node for the key value.
- A row is deleted that is the last remaining row with a key value in the same index node and that key value appeared in the index node before the previously deleted row.
- Another row is deleted with the first key value and the deleted dbkey is the last remaining dbkey in the first overflow node.

When the last dbkey is deleted from the first overflow node, that node is deleted and the overflow pointer in the index node must be modified to point to the second overflow node. In the sequence above, the deletion of the unique key values caused the location of the second ikey to be moved in the index node but the third delete used a stale pointer to update the overflow dbkey.

The problem can be avoided by performing the sequence of operations in a different order or in separate transactions. The problem only affects indexes of *type is sorted ranked*.

If index corruption occurs, the index must be dropped and recreated to eliminate the corruption.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.16 Wrong Results Generated by Query With Common Boolean Elements

Bug 4332115

In prior releases of Oracle Rdb, an optimization was applied to WHERE clauses and other Boolean expressions to reformat those queries to gain possible advantages in the query execution phase. However, in the reported problem, this optimization leads to a query strategy that does not return the correct results. In some cases, the common expression in an OR expression was lifted too high in the expression and so distorted the results.

While this problem is possible in older versions of Rdb, it may occur more frequently in Oracle Rdb V7.0 and later versions because of a more aggressive restructuring algorithm employed by these recent versions.

For example, this query against the EMPLOYEES table should produce four result rows.

```
SQL> set flags 'strategy,detail';
SQL>
SQL> select last_name, first_name
cont> from employees
cont> where
cont>     (
cont>     (
cont>         (last_name = 'Watters' and first_name = 'Christine')
```

```

cont>      or
cont>      (last_name = 'Watters' and first_name = 'Cora')
cont>      )
cont>      and
cont>      (
cont>      (last_name = 'Watters' and first_name = 'Christine')
cont>      or
cont>      (last_name = 'Watters' and first_name = 'Cora')
cont>      )
cont>      )
cont>      or
cont>      (last_name = 'Smith')
cont> order by 1, 2
cont> ;
Tables:
  0 = EMPLOYEES
Sort: 0.LAST_NAME(a), 0.FIRST_NAME(a)
Leaf#01 BgrOnly 0:EMPLOYEES Card=100
  Bool: (((0.FIRST_NAME = 'Christine') OR (0.FIRST_NAME = 'Cora')) AND ((
    0.FIRST_NAME = 'Christine') OR (0.FIRST_NAME = 'Cora'))) OR (0.LAST_NAME
    = 'Smith')) AND (0.LAST_NAME = 'Watters')
  BgrNdx1 EMP_LAST_NAME [1:1] Fan=12
  Keys: 0.LAST_NAME = 'Watters'
LAST_NAME      FIRST_NAME
Watters        Christine
Watters        Cora
2 rows selected
SQL> -- expecting 4 rows

```

The detailed strategy output shows that the expression *AND (0.LAST\_NAME = 'Watters')* has been raised to the outer most part of the query and thus erroneously eliminates two of the rows matching 'Smith'.

This problem has been corrected in Oracle Rdb Release 7.1.4.1. The query optimizer now correctly handles the case where a trailing OR term does not match a common Boolean with the query.

## 4.1.17 Wrong Result From Query With Common Join Booleans in OR

Bugs 4332115 and 1329838

The following query with common join booleans in the OR predicate returns the wrong result (should be 5 rows) after the common boolean optimization is manually disabled (by commenting out the code).

```

sel  e.employee_id, e.last_name, j.job_start
  from employees e, job_history j
  where
    e.employee_id = j.employee_id and e.employee_id = '00222'
    or
    e.employee_id = j.employee_id and e.employee_id = '00234'
    or
    e.employee_id = j.employee_id and e.employee_id = '00345'
;
Tables:
  0 = EMPLOYEES
  1 = JOB_HISTORY
Cross block of 2 entries
Cross block entry 1
  Get      Retrieval by index of relation 0:EMPLOYEES

```

## Oracle® Rdb for OpenVMS

```
Index name EMP_EMPLOYEE_ID [0:0]
Cross block entry 2
Conjunct:
((0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00222'))
OR
((0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00234'))
OR
((0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00345'))
OR index retrieval
Conjunct:
((0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00222'))
OR
((0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00234'))
OR index retrieval
Conjunct:
(0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00222')
Get Retrieval by index of relation 1:JOB_HISTORY
Index name JOB_HISTORY_HASH [1:1]
Keys: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
Conjunct:
NOT ((0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00222'))
AND
(0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00234')
Get Retrieval by index of relation 1:JOB_HISTORY
Index name JOB_HISTORY_HASH [1:1]
Keys: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
Conjunct:
NOT (((0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00222'))
OR
(0.EMPLOYEE_ID = 1.EMPLOYEE_ID) ) ! Note :: <== missing '00234'
AND (0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND (0.EMPLOYEE_ID = '00345')
Get Retrieval by index of relation 1:JOB_HISTORY
Index name JOB_HISTORY_HASH [1:1]
Keys: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
E.EMPLOYEE_ID E.LAST_NAME J.JOB_START
00222 Lasch 28-Dec-1979
00222 Lasch 18-Aug-1976
00234 Robinson 20-May-1980
00234 Robinson 5-Mar-1978
4 rows selected
```

Note that the boolean '0.EMPLOYEE\_ID = '00234' is missing in the NOT predicate of the 2nd leg of the outer OR index retrieval.

There is no known workaround for this problem.

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

1. The main query joins EMPLOYEES and JOB\_HISTORY tables using EMPLOYEE\_ID.
2. The WHERE clause of the query contains two OR predicates with three operands where E.EMPLOYEE\_ID = J.EMPLOYEE\_ID is a common join boolean in each operand.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

### 4.1.18 Wrong Result Selecting From a Derived Table of UNION Clause

## Bug 4327112

The following query selects the wrong result (should be 3 rows) from a derived table of a union clause.

```

set flags 'strategy,detail';
SEL * FROM
  (SELECT FLD1, A.FLD2,
    NVL((SELECT FLD3 FROM TAB_B B WHERE B.FLD2 = A.FLD2), 'Z')
    FROM TAB_A A
  UNION
  SELECT F1, F2, F3 FROM TAB_C
  ) AS DT (F1, F2, F3)
WHERE DT.F3 = 'Z';
Tables:
  0 = TAB_A
  1 = TAB_B
  2 = TAB_C
Merge of 1 entries
Merge block entry 1
Reduce: <mapped field>, <mapped field>, <mapped field>
Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
Merge of 2 entries
Merge block entry 1
Cross block of 2 entries
  Cross block entry 1
    Get Retrieval sequentially of relation 0:TAB_A
  Cross block entry 2
    Aggregate: 0:VIA (1.FLD3)
    Conjunct: 1.FLD2 = 0.FLD2
    Get Retrieval sequentially of relation 1:TAB_B
Merge block entry 2
Conjunct: 2.F3 = 'Z'
Get Retrieval sequentially of relation 2:TAB_C
F1    F2    F3
1     A     Z
2     A     Z
3     B     1    <== should not have returned this row
4     C     Z
4 rows selected

```

In the problem query, the conjunct "DT.F3 = 'Z'" appears only in the second UNION leg but not in the first UNION leg. Without an additional conjunct at the outside of the union query, the query returns the wrong result.

As a workaround, the query works if the union legs are swapped.

```

SEL * FROM
  (
  SELECT F1, F2, F3 FROM TAB_C ! <== second leg is swapped here as first leg
  UNION
  SELECT
    FLD1, A.FLD2,
    NVL((SELECT FLD3 FROM TAB_B B WHERE B.FLD2 = A.FLD2), 'Z')
    FROM TAB_A A
  ) AS DT (F1, F2, F3)
WHERE DT.F3 = 'Z';
Tables:
  0 = TAB_C
  1 = TAB_A
  2 = TAB_B

```

```

3 = TAB_B
Conjunct: <mapped field> = 'Z'                <== See note
Merge of 1 entries
Merge block entry 1
Reduce: <mapped field>, <mapped field>, <mapped field>
Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
Merge of 2 entries
Merge block entry 1
Conjunct: 0.F3 = 'Z'
Get      Retrieval sequentially of relation 0:TAB_C
Merge block entry 2
Cross block of 3 entries
Cross block entry 1
Get      Retrieval sequentially of relation 1:TAB_A
Cross block entry 2
Aggregate: 0:VIA (2.FLD3)
Conjunct: 2.FLD2 = 1.FLD2
Get      Retrieval sequentially of relation 2:TAB_B
Cross block entry 3
Aggregate: 1:VIA (3.FLD3)
Conjunct: 3.FLD2 = 1.FLD2
Get      Retrieval sequentially of relation 3:TAB_B
F1      F2      F3
1       A       Z
2       A       Z
4       C       Z
3 rows selected

```

Note:: There is an additional conjunct "<mapped field> = 'Z'" at the top of the union query.

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

1. The main query selects from the derived table of a union clause with a filter predicate.
2. The first leg of the union clause contains a select query with a NVL function on a subselect query.
3. The second leg of the union clause contains a select query from a table.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.19 Incorrect Foreign Key Constraint Behavior on Update

Bug 4157145

A certain class of foreign key constraint would fail to detect a violation when an update was performed. Under the following conditions, an update statement that modified a primary/unique key would not result in a constraint violation if one were to exist:

- The constraint must have been defined by the SQL REFERENCES clause (making it a foreign key constraint).
- The constraint must be self-referencing. A self-referencing constraint is one in which the columns of the foreign key and the columns of the primary/unique key are in the same table.
- The update statement must have referenced one or more of the columns that make up the primary/unique key.

The following is an example of a self-referencing, foreign key constraint.

```

SQL$
create database filename test;

create table t (pk char (3), fk char (3),
               constraint pk_constraint
               primary key (pk) not deferrable);

insert into t(pk)    values ('1');
1 row inserted
insert into t(pk,fk) values ('2','2');
1 row inserted
insert into t(pk,fk) values ('3','1');
1 row inserted
commit;

alter table t
  add constraint
    constraint fk_constraint
    foreign key (fk) references t(pk) not deferrable;
commit;

select * from t order by pk;
PK      FK
1       NULL
2       2
3       1
3 rows selected
rollback;

update t set pk='9' where pk='1';
%RDB-E-INTEG_FAIL, violation of constraint FK_CONSTRAINT caused
operation to fail
-RDB-F-ON_DB, on database DISK:[DIR]TEST.RDB;

```

The third row, with values (3,1), shows a foreign key value that matches a primary key value in the first row. The update statement, which attempts to change the primary key value in that first row from 1 to 9 violates the foreign key constraint that a primary key with value 1 exist in the table.

The example shows the error message that is now reported by Oracle Rdb Release 7.1.4.1. In some prior releases, the constraint violation was not detected, no error message was reported, and the update was allowed.

For self-referencing, foreign key constraint definitions created using SQL and the REFERENCES clause and created using Oracle Rdb 7.1.4.1, the correct behavior will appear. For such constraints created by certain earlier releases of Oracle Rdb, the problem will continue to appear. To determine if a table contains erroneous data, data that should cause a constraint violation, execute the statement RMU/VERIFY/CONSTRAINT. If there is an error in the data, you should see an error message such as the one in the following example. In the example, the database file specification has been abbreviated to the word DBASE to shorten the text.

```

$ rmu/verify/constraint test.rdb
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-BGNVCONST, beginning verification of constraints for database DBASE
%RMU-W-CONSTFAIL, Verification of constraint "FK_CONSTRAINT" has failed.
%RMU-I-ENDVCONST, completed verification of constraints for database DBASE
...

```

Oracle recommends the regular use of RMU Verify to monitor the integrity of your databases.

To correct the problem in constraint definitions created using earlier releases of Oracle Rdb, first fix the database files, either by dropping and recreating the offending constraint definitions or by recreating the database files from EXPORT/IMPORT interchange files. Then, recreate any Oracle Rdb backup files for the databases (files with filename extensions of .RBF).

- For an existing Rdb database, alter the table definition to drop the foreign key constraint. Then, alter the table definition to recreate the constraint using Oracle Rdb 7.1.4.1.
- For an existing database backup file, make a new backup file from a database file in which the constraint definition has been recreated as explained in the preceding bulleted item.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.20 Bugchecks Accessing a REAL or DOUBLE PRECISION Column

Bug 4349150

If an application inserted invalid floating point data in a REAL or DOUBLE PRECISION column, Rdb would bugcheck in various ways when that data was accessed. Inserting invalid floating point data is an application bug and can happen, for example, when uninitialized host language variables are used as a source for column data in an insert.

For example, suppose a table T was defined with a REAL column named R. If an F–Float format Not–a–number (NaN) value was stored in column R, the following SQL statements demonstrate the kinds of bugcheck failures that were occurring.

Simple select:

```
SQL> select R from T;
R
%RDMS-I-BUGCHKDMP, generating bugcheck dump file XXXXX:[XXXX]SQLBUGCHK.DMP;
%SQL-F-BUGCHK, There has been a fatal error. Please contact your Oracle
support representative. SQL$EOP - 4
```

Type conversion:

```
SQL> select cast(R as varchar(15)) FROM T;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file XXXXX:[XXXX]RDSBUGCHK.DMP;
%COSI-F-UNEXPERR, unexpected system error
-SYSTEM-E-ACCVIO, access violation, reason mask=00, virtual address=
0000000000000000, PC=0000000000000000, PS=00000000
```

Arithmetic:

```
SQL> select (R + 1.0) from T
%RDMS-I-BUGCHKDMP, generating bugcheck dump file XXXXX:[XXXX]RDSBUGCHK.DMP;
%COSI-F-UNEXPERR, unexpected system error
-SYSTEM-E-ACCVIO, access violation, reason mask=00, virtual address=
0000000000000000, PC=0000000000000000, PS=00000000
```

As a workaround for this problem, update the rows to put valid floating point data in the problem columns.

This problem has been corrected in Oracle Rdb Release 7.1.4.1. In the case of a simple select (like the first example above), the output field in the display is filled with asterisks ("\*"). In the other examples where Rdb tries to convert datatypes or do arithmetic, an %RDB-E-ARITH\_EXCEPT error is returned.

## 4.1.21 Bugchecks in PSII2SPLITNODE When Using Ranked Indexes

Bug 4324725

When inserting rows into a table with indexes of *TYPE IS SORTED RANKED*, it was possible that a bugcheck could occur in the routine PSII2SPLITNODE.

The exception occurred when the 65535th row was added to a particular key value, and the dbkey was inserted into an overflow node and the level one node contained exactly one key value and had two or fewer free bytes.

The following example shows the bugcheck footprint for an index that only contains one key value.

```
COSI-F-BUGCHECK, internal consistency failure
Exception occurred at PSII2SPLITNODE + 00000390
Called from PSII2CASETOPM1 + 000006C8
Called from PSII2INSERTTREE + 000001FC
Called from RDMS$$KOD_INSERT_TREE + 00002954
```

For indexes with more than one key value, the bugcheck footprint would be slightly different.

```
COSI-F-BUGCHECK, internal consistency failure
Exception occurred at PSII2SPLITNODE + 00000390
Called from PSII2BALANCE + 00000DEC
Called from PSII2INSERTT + 00000548
Called from PSII2INSERTT + 0000042C
```

The index is not corrupt but repeated attempts to insert such a record will fail with the same exception. If the index is dropped, the row may be inserted, and the index can be rebuilt correctly. A rebuild of the index would likely eliminate the bugcheck.

The problem can be avoided by either using alternate index types or adding fields to the key value to make the index more unique.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.22 Wrong Result from UNION Query with Outer Join Leg

Bug 4392374

The following query should find one row:

```
set flags 'strategy,detail';
SELECT sec_id, busdate, tc_code
FROM (SELECT
      SP_sec_id,
      SP_busdate,
```

```

        SP_tc_code,
        SP_rlc_code,
        SP_stc_id,
        SP_part_id
FROM
  (SELECT
    SEC.sec_id,
    SEC.sec_sub_id,
    ABD.busdate,
    STD.tc_code,
    STD.rlc_code,
    STD.stc_id,
    OPA.part_id
  FROM    STD    STD,
          ABD    ABD,
          OPA    OPA,
          RLC    RLC,
          STC    STC,
          SEC    SEC
  WHERE   SEC.es_date   <= ABD.busdate
  AND     SEC.ee_date   >= ABD.busdate
  AND     SEC.inst_code = 1
  AND     STD.es_date   <= ABD.busdate
  AND     STD.ee_date   >= ABD.busdate
  AND     STD.sec_id    = SEC.sec_id
  AND     OPA.sec_id    = SEC.sec_id
  AND     STD.rlc_code  = RLC.rlc_code
  AND     RLC.es_date   <= ABD.busdate
  AND     RLC.ee_date   >= ABD.busdate
  AND     STD.stc_id    = STC.stc_id
  AND     STC.es_date   <= ABD.busdate
  AND     STC.ee_date   >= ABD.busdate
  ) AS    SP
        (SP_sec_id,
        SP_sec_sub_id,
        SP_busdate,
        SP_tc_code,
        SP_rlc_code,
        SP_stc_id,
        SP_part_id
        )
                                LEFT OUTER JOIN

  (SELECT
    DIV1.shr_id,
    DIV1.ex_div_date,
    DIV1.div_cur_code,
    SUM(DIV1.div_value)
  FROM
    DIV    DIV1,
    ABD    ABD1,
    SEC    SEC1,
    STD    STD1
  WHERE   DIV1.ex_div_date   = ABD1.busdate
  AND     DIV1.div_cur_code  = STD1.tc_code
  AND     STD1.es_date       <= ABD1.busdate
  AND     STD1.ee_date       >= ABD1.busdate
  AND     SEC1.sec_sub_id    = DIV1.shr_id
  AND     STD1.sec_id        = SEC1.sec_id
  AND     SEC1.es_date       <= ABD1.busdate
  AND     SEC1.ee_date       >= ABD1.busdate
  AND     SEC1.inst_code     = 1
  GROUP BY

```

```

    DIV1.shr_id,
    DIV1.ex_div_date,
    DIV1.div_cur_code
) AS DIVSEC
    (DIVSEC_shr_id,
    DIVSEC_ex_div_date,
    DIVSEC_div_cur_code,
    DIVSEC_div_value)
ON SP_sec_sub_id = DIVSEC_shr_id AND
   DIVSEC_div_cur_code = SP_tc_code AND
   DIVSEC_ex_div_date = SP_busdate
UNION ALL
SELECT
    SEC.sec_id,
    ABD.busdate,
    STD.tc_code,
    STD.rlc_code,
    STD.stc_id,
    OPA.part_id
FROM STD STD,
     SEC SEC,
     OPA OPA,
     ABD ABD,
     RLC RLC,
     STC STC
WHERE SEC.es_date <= ABD.busdate
AND SEC.ee_date >= ABD.busdate
AND SEC.inst_code <> 1
AND STD.es_date <= ABD.busdate
AND STD.ee_date >= ABD.busdate
AND STD.sec_id = SEC.sec_id
AND OPA.sec_id = SEC.sec_id
AND STD.rlc_code = RLC.rlc_code
AND RLC.es_date <= ABD.busdate
AND RLC.ee_date >= ABD.busdate
AND STD.stc_id = STC.stc_id
AND STC.es_date <= ABD.busdate
AND STC.ee_date >= ABD.busdate
)
as OM_STATIC_VIEW (
    sec_id,
    busdate,
    tc_code,
    rlc_code,
    stc_id,
    part_id
)
WHERE sec_id = 32978 AND busdate = '24-MAY-2005';

```

Tables:

```

0 = STD
1 = ABD
2 = OPA
3 = RLC
4 = STC
5 = SEC
6 = DIV
7 = ABD
8 = SEC
9 = STD
10 = STD
11 = SEC
12 = OPA

```

## Oracle® Rdb for OpenVMS

```
13 = ABD
14 = RLC
15 = STC
Conjunct: (<mapped field> = 32978) AND (<mapped field> = '24-MAY-2005')
Merge of 1 entries
Merge block entry 1
Merge of 2 entries
Merge block entry 1
Conjunct: 5.sec_id = 32978
Conjunct: 1.busdate = '24-MAY-2005'
Conjunct: 5.sec_id = 32978          <== See Note1
Conjunct: 1.busdate = '24-MAY-2005'
Match (Left Outer Join)          <== See Note2
Outer loop
Sort: 1.busdate(a), 0.tc_code(a),
      5.sec_sub_id(a)          <== See Note3
Merge of 1 entries
Merge block entry 1
Cross block of 6 entries
Cross block entry 1
Conjunct: 1.busdate = '24-MAY-2005'
Index only retrieval of relation 1:ABD
Index name ABD_U_PRM [1:1] Direct lookup
Keys: <mapped field> = '24-MAY-2005'
Cross block entry 2
Conjunct: 4.es_date <= 1.busdate
Conjunct: 4.ee_date >= 1.busdate
Get Retrieval by index of relation 4:STC
Index name STC_U_PRM [0:0]
Cross block entry 3
Conjunct: 0.es_date <= 1.busdate
Conjunct: 0.ee_date >= 1.busdate
Get
Retrieval by index of relation 0:STD
Index name STD_U_FRG_7 [1:1]
Keys: 0.stc_id = 4.stc_id
Cross block entry 4
Conjunct: 5.ee_date >= 1.busdate
Get Retrieval by index of relation 5:SEC <== See Note2
Index name SEC_U_FRG_7 [2:3]
Keys: (5.inst_code = 1) AND (0.sec_id =
      5.sec_id) AND (5.es_date <=
      1.busdate)
Bool: 5.sec_id = 32978
Cross block entry 5
Conjunct: 3.ee_date >= 1.busdate
Get Retrieval by index of relation 3:RLC
Index name RLC_U_PRM [1:2]
Keys: (0.rlc_code = 3.rlc_code)
      AND (3.es_date <= 1.busdate)
Cross block entry 6
Get Retrieval by index of relation 2:OPA
Index name OPA_U_PRM [1:1] Direct lookup
Keys: 2.sec_id = 5.sec_id
Inner loop
Temporary relation
Sort: 6.ex_div_date(a), 6.div_cur_code(a), 6.shr_id(a)
Merge of 1 entries
Merge block entry 1
Aggregate: 0:SUM (6.div_value)
Sort: 6.shr_id(a), 6.ex_div_date(a), 6.div_cur_code
      (a)
```

## Oracle® Rdb for OpenVMS

```
Cross block of 4 entries
Cross block entry 1
  Index only retrieval of relation 7:ABD
  Index name  ABD_U_PRM [0:0]
Cross block entry 2
  Get      Retrieval by index of relation 6:DIV
  Index name  DIV_U_ALT_1 [1:1]
  Keys: 6.ex_div_date = 7.busdate
Cross block entry 3
  Conjunct: 8. ee_date >= 7.busdate
  Conjunct: 8. inst_code = 1
  Get      Retrieval by index of relation 8:SEC
  Index name  SEC_U_FRG_8 [1:2]
  Keys: (8.sec_sub_id = 6.shr_id) AND (
  8.es_date <= 7.busdate)
Cross block entry 4
  Conjunct: (6.div_cur_code = 9.tc_code)
  AND (9. ee_date >= 7.busdate)
  Get
  Retrieval by index of relation 9:STD
  Index name  STD_U_PRM [1:2]
  Keys: (9.sec_id = 8.sec_id) AND (
  9.es_date <= 7.busdate)
Merge block entry 2
Cross block of 6 entries
Cross block entry 1
  Conjunct: 13.busdate = '24-MAY-2005'
  Index only retrieval of relation 13:ABD
  Index name  ABD_U_PRM [1:1]  Direct lookup
  Keys: <mapped field> = '24-MAY-2005'
Cross block entry 2
  Conjunct: 15.es_date <= 13.busdate
  Conjunct: 15. ee_date >= 13.busdate
  Get      Retrieval by index of relation 15:STC
  Index name  STC_U_PRM [0:0]
Cross block entry 3
  Conjunct: 10.es_date <= 13.busdate
  Conjunct: 10. ee_date >= 13.busdate
  Get      Retrieval by index of relation 10:STD
  Index name  STD_U_FRG_7 [1:1]
  Keys: 10.stc_id = 15.stc_id
Cross block entry 4
  Conjunct: 14. ee_date >= 13.busdate
  Get      Retrieval by index of relation 14:RLC
  Index name  RLC_U_PRM [1:2]
  Keys: (10.rlc_code = 14.rlc_code) AND
  (14.es_date <= 13.busdate)
Cross block entry 5
  Conjunct: (11. ee_date >= 13.busdate) AND (
  11.inst_code <> 1)
  Get      Retrieval by index of relation 11:SEC
  Index name  SEC_U_PRM [1:2]
  Keys: (10.sec_id = 11.sec_id) AND (11.es_date
  <= 13.busdate)
  Bool: 11.sec_id = 32978
Cross block entry 6
  Get      Retrieval by index of relation 12:OPA
  Index name  OPA_U_PRM [1:1]  Direct lookup
  Keys: 12.sec_id = 11.sec_id
0 rows selected

Notel:: The conjunct "5.sec_id = 32978" is generated from the predicate
```

## Oracle® Rdb for OpenVMS

"sec\_id = 32978" since sec\_id is a mapped column from the OM\_STATIC\_VIEW (a derived table) which contains the UNION query with the first leg as a left outer join and the second leg as a regular join subquery.

Note2:: The match strategy is used for the left outer join operation and thus it requires a sort node. The sort operation in Rdb always pre-loaded the records from the table into the sort buffer before the actual record retrieval.

Note3:: When the conjunct "5.sec\_id = 32978" is evaluated, the column "5.sec\_id" contains a stale value from the table 5:SEC instead of the updated value from the sort buffer. This causes the query to return FALSE even though the correct record is found.

As a workaround, the query works if the index SEC\_U\_FRG\_7 for SEC table is dropped since the query now switches from match to cross strategy for the left outer join operation without the sort node. This could also happen if SQL flags is set to 'index\_column\_group' or 'old\_cost\_model'.

The following is the strategy output of the same query after the index SEC\_U\_FRG\_7 is dropped.

```
Conjunct: (<mapped field> = 32978) AND (<mapped field> = '24-MAY-2005')
Merge of 1 entries
  Merge block entry 1
    Merge of 2 entries
      Merge block entry 1
        Conjunct: 5.sec_id = 32978                                <== See Note4
        Conjunct: 1.busdate = '24-MAY-2005'
        Cross block of 2 entries          (Left Outer Join) <== See Note4
          Cross block entry 1
            Merge of 1 entries
              Merge block entry 1
                Cross block of 6 entries
                  Cross block entry 1
                    Conjunct: 1.busdate = '24-MAY-2005'
                    Index only retrieval of relation 1:ABD
                      Index name  ABD_U_PRM [1:1]    Direct lookup
                      Keys: <mapped field> = '24-MAY-2005'
                  Cross block entry 2
                    Conjunct: (5.ee_date >= 1.busdate) AND (
                      5.inst_code = 1)
                    Get      Retrieval by index of relation 5:SEC <== See Note4
                      Index name  SEC_U_PRM [1:2]
                      Keys: (<mapped field> = 32978) AND (5.es_date <=
                        1.busdate)
                  Cross block entry 3
                    Conjunct: 4.es_date <= 1.busdate
                    Conjunct: 4.ee_date >= 1.busdate
                    Get      Retrieval by index of relation 4:STC
                      Index name  STC_U_PRM [0:0]
                  Cross block entry 4
                    Conjunct: 0.ee_date >= 1.busdate
                    Conjunct: 0.stc_id = 4.stc_id
                    Get
                    Retrieval by index of relation 0:STD
                      Index name  STD_U_PRM [1:2]
                      Keys: (0.sec_id = 5.sec_id) AND (
                        0.es_date <= 1.busdate)
                  Cross block entry 5
                    Get      Retrieval by index of relation 2:OPA
                      Index name  OPA_U_PRM [1:1]    Direct lookup
```

## Oracle® Rdb for OpenVMS

```
Keys: 2.sec_id = 5.sec_id
Cross block entry 6
  Conjunct: 3.ee_date >= 1.busdate
  Get      Retrieval by index of relation 3:RLC
  Index name  RLC_U_PRM [1:2]
  Keys: (0.rlc_code = 3.rlc_code)
        AND (3.es_date <= 1.busdate)
Cross block entry 2
  Conjunct: (5.sec_sub_id = 6.shr_id) AND (
            6.div_cur_code = 0.tc_code) AND (
            6.ex_div_date = 1.busdate)
Merge of 1 entries
  Merge block entry 1
  Aggregate: 0:SUM (6.div_value)
  Sort: 6.shr_id(a), 6.ex_div_date(a), 6.div_cur_code
        (a)
Cross block of 4 entries
  Cross block entry 1
  Index only retrieval of relation 7:ABD
  Index name  ABD_U_PRM [0:0]
  Cross block entry 2
  Conjunct: (5.sec_sub_id = 6.shr_id) AND (
            6.div_cur_code = 0.tc_code) AND
            (6.ex_div_date = 1.busdate)
  Conjunct: 6.ex_div_date = 7.busdate
  Get      Retrieval by index of relation 6:DIV
  Index name  DIV_U_ALT_1 [2:2]
  Keys: (6.ex_div_date = 7.busdate) AND (
        5.sec_sub_id = 6.shr_id)
  Cross block entry 3
  Conjunct: 8.ee_date >= 7.busdate
  Conjunct: 8.inst_code = 1
  Get      Retrieval by index of relation 8:SEC
  Index name  SEC_U_FRG_8 [1:2]
  Keys: (8.sec_sub_id = 6.shr_id) AND (
        8.es_date <= 7.busdate)
  Cross block entry 4
  Conjunct: (6.div_cur_code = 9.tc_code)
            AND (9.ee_date >= 7.busdate)
  Get
  Retrieval by index of relation 9:STD
  Index name  STD_U_PRM [1:2]
  Keys: (9.sec_id = 8.sec_id) AND (
        9.es_date <= 7.busdate)
Merge block entry 2
Cross block of 6 entries
  Cross block entry 1
  Conjunct: 13.busdate = '24-MAY-2005'
  Index only retrieval of relation 13:ABD
  Index name  ABD_U_PRM [1:1]   Direct lookup
  Keys: <mapped field> = '24-MAY-2005'
  Cross block entry 2
  Conjunct: 15.es_date <= 13.busdate
  Conjunct: 15.ee_date >= 13.busdate
  Get      Retrieval by index of relation 15:STC
  Index name  STC_U_PRM [0:0]
  Cross block entry 3
  Conjunct: 10.es_date <= 13.busdate
  Conjunct: 10.ee_date >= 13.busdate
  Get      Retrieval by index of relation 10:STD
  Index name  STD_U_FRG_7 [1:1]
  Keys: 10.stc_id = 15.stc_id
```

```

Cross block entry 4
  Conjunct: 14.ee_date >= 13.busdate
  Get      Retrieval by index of relation 14:RLC
  Index name  RLC_U_PRM [1:2]
  Keys: (10.rlc_code = 14.rlc_code) AND
        (14.es_date <= 13.busdate)
Cross block entry 5
  Conjunct: (11.ee_date >= 13.busdate) AND (
            11.inst_code <> 1)
  Get      Retrieval by index of relation 11:SEC
  Index name  SEC_U_PRM [1:2]
  Keys: (10.sec_id = 11.sec_id) AND (11.es_date
        <= 13.busdate)
  Bool: 11.sec_id = 32978
Cross block entry 6
  Get      Retrieval by index of relation 12:OPA
  Index name  OPA_U_PRM [1:1]   Direct lookup
  Keys: 12.sec_id = 11.sec_id
sec_id  busdate                tc_code
 32978  24-MAY-2005 00:00:00.00  CHF
1 row selected

```

Note4:: The cross strategy is used for the left outer join operation instead of a match strategy. Since no sort is required and the column 5.sec\_id in the conjunct "5.sec\_id = 32978" did not get pre-loaded, as in the case of a sort operation, it was correctly retrieved from the table on the fly during the cross operation.

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

1. The main query selects a particular row from either a view or derived table of a union query where the first leg is a left outer join subquery and the second leg is a regular join subquery.
2. The column of the equality predicate is mapped via the view or derived table to the same base table which is included in both legs of the left outer join subquery and the second UNION leg.
3. The query applies a match strategy for the left outer join operation with a sort node at the outer loop and another sort node at the inner loop.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.23 COSI\_MEM\_FREE\_VMLIST Bugcheck with Vertical Partitioning

Bug 4460398

With vertical partitioning in use, the following bugcheck was possible:

```

***** Exception at 0091B704 : COSI_MEM_FREE_VMLIST + 00000094
%SYSTEM-F-ACCPIO, access violation, reason mask=04,
  virtual address=00000000008308D, PC=000000000091B704, PS=0000001B

```

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.24 Bugcheck from INSERT With Partition Index

Bug 4477862

The following query bugchecks:

```
SQL>INSERT INTO TBL1 VALUES (1,2,3);
%DEBUG-I-DYNMODSET, setting module RDMS$PREEXEMSC
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=000000000000
0060, PC=000000000006A2358, PS=00000018
1816:                                IF .CXPR [CXPR$L_OPERATOR] NEQU BLR$K_MISSING
```

where the tables are defined as follows:

```
CREATE TABLE TBL1 (TBL1_P1 INT, TBL1_P2 INT, TBL1_P3 INT, CONSTRAINT TBL1_PRIM
PRIMARY KEY (TBL1_P1, TBL1_P2) DEFER);
```

```
CREATE TABLE TBL2 (TBL2_P1 INT, TBL2_P2 INT, TBL2_P3 INT, CONSTRAINT TBL2_FOR
FOREIGN KEY (TBL2_P1, TBL2_P2) REFERENCES TBL1 (TBL1_P1, TBL1_P2) DEFER);
```

```
CREATE INDEX IDX1 ON TBL1 (TBL1_P1 DESC, TBL1_P2 DESC) STORE USING (TBL1_P1)
IN A1 WITH LIMIT OF (2)
IN A2 WITH LIMIT OF (1)
OTHERWISE IN A3;
CREATE INDEX IDX2 ON TBL2 (TBL2_P1, TBL2_P2) ;
COMMIT;
```

```
UPDATE RDB$RELATIONS SET RDB$CARDINALITY = 306
WHERE RDB$RELATION_NAME='TBL2' ;
COMMIT;
```

As a workaround, the query works if the cardinality of table TBL2 is updated to a number below 306 rows.

```
UPDATE RDB$RELATIONS SET RDB$CARDINALITY = 305
WHERE RDB$RELATION_NAME='TBL2' ;
COMMIT;
```

This problem with the INSERT statement occurs when the following conditions are met:

1. The table TBL1 contains a primary constraint and the table TBL2 contains a foreign key constraint referencing the primary keys of TBL1.
2. The index for the table TBL1 is a partition index keying on the leading primary keys.
3. The cardinality of the table TBL2 is updated to a number such that the optimizer chooses a match strategy over cross. In this case, the cardinality number is 306.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.25 Journals Not Initialized After Backup if Backing Up to Tape Device

Bug 2808539

If after image journal backups were being done to tape and the AIJs being backed up were circular AIJs, the

backed up journal would not get properly initialized. This could lead to various issues such as journaling being shutdown with a "journal is not empty" error. When that occurred, the journal state displayed by `RMU/DUMP/HEADER=JOURNAL` would show the following lines of output:

```
File is inaccessible
  journal has been made inaccessible by system
  journal is not empty
```

Other symptoms were also possible. The database recovery process (DBR) could fail with a bugcheck in `DBR$RECOVER_ALL`. Attempts to recover a database using an existing journal that was not properly re-initialized could fail with `AIJCORRUPT` errors. For example:

```
%RMU-W-AIJCORRUPT, journal entry 1451795/3364123 contains a new AIJBL that
doesn't have the start flag set
```

This problem was introduced in Oracle Rdb Release 7.1.2.3.

The problem can be avoided by backing up the journals to a disk destination.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.26 Constant Snapshot File Growth

With Oracle Rdb Release 7.1.4, it was possible for snapshot files to continually extend. This would occur after an abnormal user termination was handled by a database recovery (DBR) process. The problem was seen after installing Oracle Rdb Release 7.1.4.

There are two data structures in the database root (`.RDB`) file that are used to represent the state of a database user:

1. The RTUPB list
2. The TSNBLKs

In Release 7.1.4, when the DBR process would clear the failed user's entries in the root file, it would neglect to clear the TSNBLK entry. That would cause Oracle Rdb to include the old user's last transaction sequence number (TSN) when determining what TSN should be granted to new snapshot (read only) transactions. Typically, the TSNBLK entry would soon get reused by another user and no symptoms of the problem would be seen. Occasionally, it was possible for the old TSNBLK entry to linger for quite some time. As long as the old TSN was still in the TSNBLK, pages in the snapshot files with TSNs greater than the old user's TSN could not be reclaimed. This would cause the snapshot files to grow since pages in the file could not be reused.

To determine if this problem is affecting a database, the following steps can be done:

1. Determine the TSN being granted to all snapshot transactions:

```
$ RMU/DUMP/HEADER/OPTION=DEBUG/OUTPUT=TEMP.TXT DB
$ SEARCH TEMP.TXT /WINDOW=(0,3) "Snapshot transaction in progress"
  Snapshot transaction in progress
  Last Process quiet-point was AIJ sequence 0
  Transaction sequence number is 0:3392
```

Note that in the above example the TSN is 3392.

2. See if there is a TSNBLK entry showing an active transaction with that TSN:

```
$ SEARCH TEMP.TXT 3392
    Transaction sequence number is 0:3392
SLOT[1.] SIP TSN = 0:3392, COMMIT_TSN = 0:0.
SLOT[2.] WIP TSN = 0:3392, COMMIT_TSN = 0:3390.
```

Note that in the above example there is a line that contains "WIP TSN = 0:3392". This indicates that there should be a user with a read write transaction with sequence number 3392.

3. Confirm that there are no read–write transactions active with that TSN:

```
$ PIPE SEARCH /WINDOW=(0,3) TEMP.TXT "Read/write transaction in progress" | -
SEARCH SYS$INPUT "0:3392"
%SEARCH-I-NOMATCHES, no strings matched
```

No read write transactions exist with that TSN, thus the TSNBLK contains an invalid entry.

This problem can be corrected by forcing the TSNBLK entry to get reused. That can be done by adding multiple concurrent attaches to the database until the TSNBLK entry is reclaimed by one of the new users. For example:

```
SQL> ATTACH 'ALIAS A1 FILENAME DB';
SQL> ATTACH 'ALIAS A2 FILENAME DB';
SQL> ATTACH 'ALIAS A3 FILENAME DB';
SQL> ATTACH 'ALIAS A4 FILENAME DB';
...
```

Since each TSNBLK can only be used by one node in the cluster, it may be necessary to do attaches from each node that currently has the database open. Continue to add attaches to the database until the "WIP TSN =" string noted above is no longer found in the RMU/DUMP/HEADER output.

This problem has been corrected in Oracle Rdb Release 7.1.4.1. The TSNBLK is now properly cleared by the DBR when recovering a failed user.

## 4.1.27 Select Count Query with Host Variable Returns Wrong Result

Bug 4208119

The following select count query, which contains a WHERE clause with a host variable, returns the wrong result.

```
declare :x integer;
begin
set :x = 1;
set :x = 1;
end;

sel count(*) from employees where :x = 2;
```

```

Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (*)
Index only retrieval of relation 0:EMPLOYEES
  Index name  EMP_EMPLOYEE_ID [0:0]

          100
1 row selected

```

The query works if the host variable is removed, as in the following example.

```

sel count(*) from employees where 1 = 2;
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (*)
Conjunct: 1 = 2
Index only retrieval of relation 0:EMPLOYEES
  Index name  EMP_EMPLOYEE_ID [0:0]

          0
1 row selected

```

The query also works if the count is removed, as in the following example.

```

sel * from employees where :x = 2;
Tables:
  0 = EMPLOYEES
Leaf#01 FFirst 0:EMPLOYEES Card=10000
  Bool: <var0> = 2
  BgrNdx1 EMP_EMPLOYEE_ID [0:0] Fan=17
  Bool: <var0> = 2
0 rows selected

```

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.28 UNION Join Query with Host Variable in the Predicate Returns Wrong Result

Bug 4208119

The following UNION join query with host variable in the predicate returns the wrong result.

```

DECLARE  :H_BID INTEGER;
DECLARE  :H_CUST VARCHAR(5);
BEGIN
SET :H_BID = 1;
SET :H_CUST = 'USCC';
END;

SELECT T1.BID, T1.CTY
FROM T1 INNER JOIN (
  SELECT CTY, MRK FROM T2
  WHERE MRK IN (2, 20)
  UNION
  SELECT CTY, MRK FROM T2
  WHERE MRK NOT IN (2, 20)
) AS T2

```

## Oracle® Rdb for OpenVMS

```

        ON T1.CTY = S.CTY AND T1.MRK = S.MRK
WHERE   T1.BID = :H_BID
        AND (T1.CUSTOMER = :H_CUST
            OR :H_CUST = '' ) ;          <== see Note

Tables:
  0 = T1
  1 = T2
  2 = T2
Cross block of 2 entries
Cross block entry 1
  Leaf#01 FFirst 0:T1 Card=1
    Bool: (0.BID = <var0>) AND ((0.CUSTOMER = <var1>) OR (<var2> =
      ''))
    BgrNdx1 T1_IDX01 [1:1] Fan=15
    Keys: 0.BID = <var0>
Cross block entry 2
  Conjunct: (0.CTY = <mapped field>) AND (0.MRK =
    <mapped field>)
Merge of 1 entries
Merge block entry 1
  Reduce: <mapped field>, <mapped field>
  Sort: <mapped field>(a), <mapped field>(a)
Merge of 2 entries
Merge block entry 1
  Conjunct: (1.MRK = 2) OR (1.MRK = 20)
  Index only retrieval of relation 1:T2
    Index name T2_IDX01 [0:0]
Merge block entry 2
  Conjunct: <var2> = ''          <== see Note
  Conjunct: 0.BID = <var0>
  Conjunct: (2.MRK <> 2) AND (2.MRK <> 20)
  Index only retrieval of relation 2:T2
    Index name T2_IDX01 [0:0]
0 rows selected

```

Note:: The conjunct ":H\_CUST = "" is pushed down to the second leg of the UNION clause. Since this is the right side operand of the OR predicate it cannot be stripped out and pushed down without the other operand.

The query works if the host variable in one of the predicates is replaced by the text literal 'USCC', as in the following example.

```

SELECT T1.BID, T1.CTY
FROM T1 INNER JOIN (
    SELECT CTY, MRK FROM T2
    WHERE MRK IN (2, 20)
    UNION
    SELECT CTY, MRK FROM T2
    WHERE MRK NOT IN (2, 20)
) AS T2
ON T1.CTY = S.CTY AND T1.MRK = S.MRK
WHERE   T1.BID = :H_BID
        AND (T1.CUSTOMER = :H_CUST
            OR 'USCC' = ''          ! <== replaced with 'USCC'
        ) ;

Tables:
  0 = T1
  1 = T2
  2 = T2
Cross block of 2 entries
Cross block entry 1
  Leaf#01 FFirst 0:T1 Card=1

```

## Oracle® Rdb for OpenVMS

```
Bool: (0.BID = <var0>) AND ((0.CUSTOMER = <var1>) OR ('USCC' =
    ''))
BgrNdx1 T1_IDX01 [1:1] Fan=15
Keys: 0.BID = <var0>
Cross block entry 2
Conjunct: (0.CTY = <mapped field>) AND (0.MRK =
    <mapped field>)
Merge of 1 entries
Merge block entry 1
Reduce: <mapped field>, <mapped field>
Sort: <mapped field>(a), <mapped field>(a)
Merge of 2 entries
Merge block entry 1
Conjunct: (1.MRK = 2) OR (1.MRK = 20)
Index only retrieval of relation 1:T2
Index name T2_IDX01 [0:0]
Merge block entry 2
Conjunct: 0.BID = <var0> <== see Note1
Conjunct: (2.MRK <> 2) AND (2.MRK <> 20)
Index only retrieval of relation 2:T2
Index name T2_IDX01 [0:0]
T1.BID T1.CTY
1 12
1 row selected
```

Note1:: Note that the conjunct "'USCC' = ''" does not appear in the second leg of the UNION clause.

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

1. The main query selects from inner-joining a table with a UNION of two sub-select queries.
2. The WHERE clause contains an OR predicate referencing a host variable twice in the equality filters.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

### 4.1.29 Bugcheck in COSI\_MEM\_FREE\_VMLIST After Update of Ranked Index

Bug 4191015

During updates on indexes of *TYPE IS SORTED RANKED*, it was possible that memory could be overwritten and memory queues become corrupt. When the memory queue was subsequently processed, a bugcheck would result.

In the following example, the update statement left a memory queue in this corrupt state. During exit from SQL, while detaching from the database, the memory queue was processed to free up the memory and a bugcheck resulted.

```
SQL> update some_table set some_key=626 where another_key=624;
32419 rows updated
SQL> commit;
SQL> exit
%RDMS-I-BUGCHKDMP, generating bugcheck dump file MBRADLEY_USR:[BRADLEY]
```

RDSBUGCHK.DMP;

The following example shows the exception and the first few calls from the bugcheck.

```
%SYSTEM-F-ACCVIO, access violation, reason mask=04, virtual
address=0000000050000004, PC=00000000011D68E4, PS=0000000B

Saved PC = 0115C214 : RDMS$$RDMSHEMA_UNLOAD_META + 00000434
Saved PC = 0115CB5C : RDMS$$RDMSHEMA_UNLOAD_TABLE + 0000024C
Saved PC = 0117FCD4 : RDMS$$DETACH_DATABASE + 000002F4
Saved PC = 0117F944 : RDMS$TOP_DETACH_DATABASE + 00000424
```

In the reported case, the update completed successfully and the index was not corrupt in any way.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.1.30 ILLPAGCNT Exception Reading Large Table with a Dynamic Tactic

Bug 3194130

When the dynamic optimizer was chosen for a query and more than 268,435,455 dbkeys were read from a single index scan, the query would fail with an illegal page count parameter exception.

The same problem also produced a bugcheck dump in versions prior to 7.1.

An example of this is shown below.

```
SQL> select * from t1 where f1>1 and f2>1 optimize for total time;
Leaf#01 BgrOnly T1 Card=300000000
  BgrNdx1 I1 [1:0] Fan=308
  BgrNdx2 I2 [1:0] Fan=308
%COSI-F-VASFULL, virtual address space full
-SYSTEM-F-ILLPAGCNT, illegal page count parameter
SQL>
```

The problem can be avoided by disabling the dynamic optimizer by either using a query outline with *EXECUTION OPTIONS NONE* or using the *MAX\_STABILITY* flag.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.2 SQL Errors Fixed

### 4.2.1 SQL Precompiler (Pascal) Generates Incorrect Definition for SQL\_TINYINT Type

Bug 4121870

In Oracle Rdb Release 7.1.3, the Pascal support in the SQL Precompiler was modified to make better use of recent OpenVMS Pascal compiler enhancements. Unfortunately, the SQL\_TINYINT definition is now no longer sharable when using multiple SQL Precompiler modules in a Pascal Environment file.

The following example shows the error that is generated.

```
$ SQL$PRE /PASCAL PRE_COMP_TEST_MAIN

  SQL_TINYINT = SQL_BYTE;
.^
%PASCAL-E-REDECL, A declaration of SQL_TINYINT already exists in environment
TEST$:PRE_COMP_TEST_MODULE.PEN
at line number 28 in file DISK12:[TESTER.SQL$PRE]PRE_COMP_TEST_MAIN.PAS;1
%PASCAL-E-ENDDIAGS, PASCAL completed with 1 diagnostic
```

This problem has been corrected in Oracle Rdb Release 7.1.4.1. The Pascal [HIDDEN] attribute is now applied to SQL\_TINYINT.

### 4.2.2 LOCK TABLE May Bugcheck if DROP TABLE Appears in Same Transaction

Bug 4134139

In prior versions of Oracle Rdb, a SET TRANSACTION ... RESERVING of a table followed by a DROP of the table referenced by the RESERVING clause and then a LOCK TABLE of any table in this same transaction might generate a bugcheck dump. The following example shows the problem.

```
SQL> set transaction read write wait reserving TAB1 for shared write;
SQL> lock table TAB1 for shared write mode wait;
SQL> rollback;
SQL> --
SQL> set transaction read write wait reserving TAB1 for shared write;
SQL> drop table TAB1;
SQL> create table TAB2 (col1 integer, col2 integer);
SQL> lock table TAB2 for shared write mode wait;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTER]RDSBUGCHK.DMP;
%RDB-F-BUG_CHECK, internal consistency check failed
SQL> rollback;
```

This problem has been corrected in Oracle Rdb Release 7.1.4.1. LOCK TABLE now correctly processes the reserving list of the transaction.

## 4.2.3 CREATE VIEW May Fail With a "Deadlock on Client" Error

Bug 4101404

In prior versions of Oracle Rdb, it was possible in rare cases for the CREATE VIEW statement to fail with a DEADLOCK error. This could occur under the following circumstances:

- The database is active with views created and dropped often. Over a long period, this will result in the unique identifier for the view growing to the maximum supported by Rdb.
- Subsequent CREATE VIEW statements must scan the RDB\$RELATIONS system table looking for unused ID's, that is, those that are freed by a DROP VIEW.
- The view being defined references a view or table with COMPUTED BY columns that also include table references (subselects).

The following example shows the problem.

```
SQL> create view DEPT_STATS as
cont>  select DEPARTMENT_NAME, EMPTY_DEPARTMENT
cont>  from DEPARTMENTS;
%RDB-E-DEADLOCK, request failed due to resource deadlock
-RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-DEADLOCK, deadlock on client '.....0..' 0000300B0000000400000055
SQL>
```

In this example, the computed column EMPTY\_DEPARTMENT references another COMPUTED BY column in DEPARTMENTS that references a different table.

This problem has been corrected in Oracle Rdb Release 7.1.4.1. Rdb now correctly handles the inner references to other tables during the locking of the view ID.

## 4.2.4 TRUNCATE TABLE Did Not Release Strong Lock on Table Until DISCONNECT

Bug 3627324

The TRUNCATE TABLE statement did not correctly queue a commit/rollback action to release locks for the target table in all cases. However, if the table had SORTED or SORTED RANKED indices then this action was queued. Otherwise, the lock on the table was retained until DISCONNECT time and prevented access to the truncated table.

This problem has been corrected in Oracle Rdb Release 7.1.4.1. COMMIT and ROLLBACK following a TRUNCATE table now release the strong metadata lock on the table to allow sharing of the table with other applications. Please note that TRUNCATE TABLE requires exclusive access to the table while it changes metadata and data in the table and indices.

## 4.2.5 COMMENT ON COLUMN Failed When Applied to a View Definition

In prior versions of Oracle Rdb 7.1, the COMMENT ON TABLE statement would apply a comment to a view and was equivalent to the COMMENT ON VIEW statement. However, similar table related COMMENT ON statements would fail with RDMS-F-NOCHGVW errors if the target was a view instead of a table.

The following example shows the errors reported by these commands.

```
SQL> comment on column current_job.JOB_START is '';
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-NOCHGVW, the definition of a view may not be changed
SQL> comment on current_job
cont>      (LAST_NAME is 'last name',
cont>      FIRST_NAME is 'first name',
cont>      JOB_START is 'job start');
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-NOCHGVW, the definition of a view may not be changed
```

This problem has been corrected in Oracle Rdb Release 7.1.4.1. COMMENT ON COLUMN and COMMENT ON with a list of column names can be used for a table or a view definition.

## 4.2.6 Unexpected SQL-F-CURALROPE Following Compound Statement or CALL Statement

Bug 4301488

In Oracle Rdb Release 7.1.4, a problem was introduced that could cause SQL to believe a cursor was still open after a COMMIT or ROLLBACK was executed in a stored procedure (activated by the CALL statement) or in a compound statement (BEGIN ... END).

The following example shows the unexpected error.

```
SQL> declare cursor2 read only table cursor for
cont>      select college_name from colleges where college_code = 'BATE';
SQL> open cursor2;
SQL> begin commit; end;
SQL> open cursor2;
SQL> begin commit; end;
SQL> open cursor2;
%SQL-F-CURALROPE, Cursor CURSOR2 was already open
```

Clearly the COMMIT statement closed the cursor but SQL did not detect that change of state. This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.2.7 Unexpected Results When Using Host Variables in Subselects

Bug 2316744

In prior versions of Oracle Rdb, assignments to host variables might not be used when subselects use these host variables inside conditional or control statements (IF statement, CASE statement or FOR cursor loop).

Host variables are those declared in C, COBOL, FORTRAN, etc in a SQL\$PRE source module or in Interactive and Dynamic SQL using the DECLARE Variable statement.

The following example shows the problem. The query should return the LAST\_NAME matching the EMPLOYEE\_ID assigned to the host variable :OUT\_VAL. However, Oracle Rdb does not detect that the subselect uses a variable modified within the scope of the IF statement and promotes the subselect into the query for the IF statement. That is, Rdb evaluates the WHERE clause prior to the assignment of the value in the IF body. The result is no matching rows found and a NULL (shows as -1 for the indicator variable) result.

```
SQL> declare :out_val char(5) = '';
SQL> declare :out_name varchar(40) = '';
SQL> declare :out_name_ind integer = 0;
SQL>
SQL> begin
cont> if exists (select count(*)
cont>             from employees
cont>             where employee_id = '00164')
cont> then
cont>     set :out_val = '00164';
cont>     set :out_name indicator :out_name_ind =
cont>         (select trim(last_name)
cont>          from employees
cont>          where employee_id = :out_val);
cont> end if;
cont> end;
SQL>
SQL> print :out_val, :out_name, :out_name_ind;
OUT_VAL  OUT_NAME                                OUT_NAME_IND
00164                                         -1
SQL>
```

Workarounds include: assigning the values prior to the conditional statement; assigning to local variables (the DECLARE appears inside the compound statement); and possibly assigning the results to the host variable after the conditional statement.

---

#### Note

*This problem does not exist for locally declared variables or parameters to stored procedures and stored functions.*

---

This problem has been corrected in Oracle Rdb Release 7.1.4.1. Oracle Rdb now detects the update of the host variable and correctly processes the subselect.

## 4.2.8 Unexpected Bugcheck Reported When LIST OF BYTE VARYING Column has NOT NULL Constraint

Bug 4354994

In prior versions of Oracle Rdb, defining a column of LIST OF BYTE VARYING, LONG or LONG RAW with a NOT NULL constraint would cause both DROP TABLE and TRUNCATE TABLE to fail with a bugcheck. In addition, if an UPDATE statement assigned NULL to a column of these types, a similar bugcheck was generated.

The following example shows the problem.

```
SQL> create table TEST_A
cont>     (col_a integer
```

```

cont>      ,col_b long raw
cont>      not null deferrable);
SQL>
SQL> drop table TEST_A;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTER]RDSBUGCHK.DMP;
%RDB-F-BUG_CHECK, internal consistency check failed
SQL>

```

The bugcheck dump summary is similar to this example:

- COSI-F-BUGCHECK, internal consistency failure
- Exception occurred at RDMS\$\$DTYPE\_BLR + 00000D74
- Called from RDMS\$\$VERIFY\_TABLE\_CONSTRAINTS + 00002294
- Called from RDMS\$\$CREATE\_ECON + 00000618
- Called from RDMS\$\$CREATE\_EMOD + 000019F4

This problem has been corrected in Oracle Rdb Release 7.1.4.1. The NOT NULL constraint is now correctly handled by UPDATE, DROP TABLE and TRUNCATE TABLE.

## 4.2.9 DEFAULT Expression Fails for Declared Temporary Tables

Bug 4390079

In prior releases of Oracle Rdb V7.1, the use of the DEFAULT clause in an INSERT or an UPDATE of a declared local temporary table might result in an error message or in the assignment of an incorrect default value.

- If no created table or view exists with the same name as the declared temporary table, then an error will be reported during the INSERT or UPDATE statement:

```

%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-RELNOEXI, relation LOCAL_TABLE0 does not exist in this database

```

- If a created table or view exists with the same name as the declared temporary table but with no matching column, then an error will be reported during the INSERT or UPDATE statement:

```

%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-BAD_SYM, unknown field symbol - MCOL2

```

- If a created table or view exists with the same name as the declared temporary table and with a matching column name, then an error may be reported during the INSERT or UPDATE statement if the data type of the DEFAULT is incompatible with the column in the temporary table.

```

%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-COSI-F-INPCONERR, input conversion error

```

If the types are compatible, then the wrong DEFAULT may be assigned to the temporary table.

The following example shows the reported error:

```

SQL> declare local temporary table module.local_table0
cont>      (mcol1 char(7) default 'o--0--o'
cont>      ,mcol2 integer default 4044

```

```

cont>    );
SQL>
SQL> insert into module.local_table0 values ('qqq', default);
%RDB-E-OBSOLETE_METADATA, request references metadata objects that no longer exist
-RDMS-F-RELNOEXI, relation LOCAL_TABLE0 does not exist in this database

```

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.2.10 DEFAULT Inherited from Domain now Displayed by SHOW TABLE

Bug 4424589

In prior releases of Oracle Rdb, the SHOW TABLE (COLUMN) output did not display the DEFAULT inherited from domains used for the column definition.

If the domain DEFAULT is overridden by a column level definition, then it is this DEFAULT that is used by Oracle Rdb and shown by the SHOW TABLE command. However, if no local definition is used then the domain's DEFAULT is inherited by the column. This value is now displayed by the SHOW TABLE command.

```

SQL> create domain MONEY integer(2) default 0.00;
SQL>
SQL> create table EXPENSES
cont>    (descr          varchar(40)
cont>    ,amt MONEY
cont>    ,amt2 MONEY default 1.11
cont>    );
SQL>
SQL> show table (column) EXPENSES
Information for table EXPENSES

Columns for table EXPENSES:
Column Name          Data Type          Domain
-----
DESCR                VARCHAR(40)
AMT                  INTEGER(2)         MONEY
  Oracle Rdb default: 0
AMT2                  INTEGER(2)         MONEY
  Oracle Rdb default: 1.11
SQL>

```

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.2.11 Dynamic SQL Rounds Results from Division Operator

Bug 4165206

In previous versions of Oracle Rdb, the Dynamic SQL interface would determine the result data type using the type and scale of the dividend. This led to rounded results if the dividend was a fixed point value. With this release of Oracle Rdb, all numeric division computations return a REAL or DOUBLE PRECISION value.

The following example shows the behavior in prior versions.

```

$ r test$tools:tester
Enter statement:
attach 'filename sql$database';
Enter statement:
select 1/2 from rdb$database;
0/: 1
Enter statement:

```

In this example the result (0.5) is rounded to the target type of INTEGER to yield a – possibly unexpected – value 1. Now Dynamic SQL uses a floating result and gives the expected result.

```

$ r test$tools:tester
Enter statement:
attach 'filename sql$database';
Enter statement:
select 1/2 from rdb$database;
0/: 0.500000
Enter statement:

```

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.2.12 SQL Incorrectly Truncated Multi–octet Characters when Using GB18030 and UTF8 Character Sets

Bug 4319811

In previous versions of Oracle Rdb, SQL would incorrectly determine that multi–octet characters from the GB18030 or UTF8 character sets were truncated on text copies when they were not actually truncated. As a result of this incorrect determination, SQL would replace the last 2 or 3 octets of a multi–octet character with underscore characters ( "\_").

In the following example, the value in hexadecimal is a valid 4 octet GB18030 character. However when a substring of the first character is extracted by SQL, the character is incorrectly truncated and the last two octets of the character are replaced with underscores.

```

SQL> set character length 'characters';
SQL> select substring (_GB18030 X'8139D531' from 1 for 1 ) from rdb$database;

9__
1 row selected

```

The correct output should have been:

```

SQL> select substring (_GB18030 X'8139D531' from 1 for 1 ) from rdb$database;

901
1 row selected

```

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.2.13 New COMMIT EVERY Clause Added to IMPORT DATABASE Command

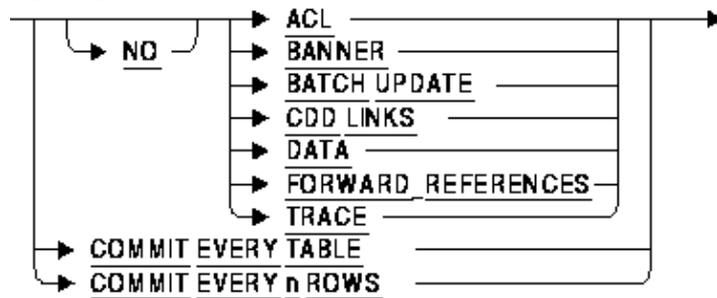
## Bug 4001638

In previous versions of Oracle Rdb, the IMPORT DATABASE statement would import and load data rows in a single transaction. If the table was very large, the recovery unit journal (RUJ) file might become quite large. To help reduce the RUJ file usage during IMPORT, a new COMMIT EVERY n ROWS clause has been added. This clause instructs SQL IMPORT to commit at regular intervals. If the IMPORT should subsequently fail, then the table will be left with a partial set of rows. The default is COMMIT EVERY TABLE.

## Format

---

Import-options=




---

The following example shows the use of this clause.

```

SQL> import database
cont>   from 'TEST$DB_SOURCE:MF_PERSONNEL'
cont>   filename 'MF_PERSONNEL'
cont>
cont>   commit every 10 rows
cont>
cont>   create storage area DEPARTMENTS
cont>     filename 'DEPARTMENTS'
cont>     page format is mixed
cont>     snapshot filename 'DEPARTMENTS'
cont>   create storage area EMPIDS_LOW
cont>     filename 'EMPIDS_LOW'
cont>     page format is mixed
cont>     snapshot filename 'EMPIDS_LOW'
cont>   create storage area EMPIDS_MID
cont>     filename 'EMPIDS_MID'
cont>     page format is mixed
cont>     snapshot filename 'EMPIDS_MID'
cont>   create storage area EMPIDS_OVER
cont>     filename 'EMPIDS_OVER'
cont>     page format is mixed
cont>     snapshot filename 'EMPIDS_OVER'
...
cont> ; ! end of import
Definition of STORAGE AREA RDB$SYSTEM overridden
Definition of STORAGE AREA MF_PERS_SEGSTR overridden
Definition of STORAGE AREA EMPIDS_LOW overridden
Definition of STORAGE AREA EMPIDS_MID overridden
Definition of STORAGE AREA EMPIDS_OVER overridden
Definition of STORAGE AREA DEPARTMENTS overridden
  
```

Definition of STORAGE AREA SALARY\_HISTORY overridden  
 Definition of STORAGE AREA JOBS overridden  
 Definition of STORAGE AREA EMP\_INFO overridden  
 COMMIT EVERY ignored for table EMPLOYEES due to PLACEMENT VIA INDEX processing  
 COMMIT EVERY ignored for table JOB\_HISTORY due to PLACEMENT VIA INDEX processing  
 SQL>

Note

*If the table being imported includes a storage map with the "PLACEMENT VIA INDEX" clause, then the COMMIT EVERY clause is ignored for that table. This restriction will be removed in a future release of Oracle Rdb. A message will be displayed informing the database administrator of the tables that did not have COMMIT EVERY applied. See the example in this release note which shows the message reported for the EMPLOYEES and JOB\_HISTORY tables.*

## 4.2.14 Unexpected SQL-F-BADBLOB Reported by SQL IMPORT

Bug 4001638

In prior releases of Oracle Rdb, it was possible to receive the following errors when importing large tables (multi-million rows) that contained one or more LIST OF BYTE VARYING columns.

```
...
IMPORTing table LARGE_TABLE
%SQL-F-BADBLOB, unable to import a list
%COSI-F-EXQUOTA, exceeded quota
-SYSTEM-F-EXQUOTA, process quota exceeded
%SQL-F-BADBLOB, unable to import a list
%RDB-E-BAD_SEGSTR_HAND, invalid segmented string handle
%RDB-E-BAD_SEGSTR_HAND, invalid segmented string handle

...
IMPORTing table LARGE_TABLE
%SQL-F-BADBLOB, unable to import a list
%COSI-F-UNEXPERR, unexpected system error
-SYSTEM-F-ILLPAGCNT, illegal page count parameter
%SQL-F-BADBLOB, unable to import a list
%RDB-E-BAD_SEGSTR_HAND, invalid segmented string handle
%RDB-E-BAD_SEGSTR_HAND, invalid segmented string handle
```

This error occurs due to the growth of a data structure that is sized to contain references to all created LIST OF BYTE VARYING columns. This data structure is used to provide the full generality of the LIST cursor model, however, during IMPORT this functionality is never used.

This problem has been corrected in Oracle Rdb Release 7.1.4.1. Oracle Rdb now releases this data structure after the row INSERT during an IMPORT operation to prevent this error and also reduce page faulting during a successful IMPORT.

A possible workaround is to define the logical name RDMS\$BIND\_SEGMENTED\_STRING\_COUNT to a value that is calculated using this formula: row-count \* list-columns + 100. The data structure is normally

created small and grows as more LIST OF BYTE VARYING data is inserted. This logical name allows the structure to be pre-created and therefore use less virtual memory. However, this pre-allocated size might still exhaust the page file quota for the process.

## 4.2.15 Connection Name Longer than 31 Octets Mishandled

Bug 4380993

Starting in Oracle Rdb Release 7.1.4, the 31 octet name limit for connection names began to be enforced. Unfortunately, if a connection name was submitted which had greater than the allowed length of 31 octets, the name would be silently truncated. This has been corrected so that the CONNECT statement is rejected with a %SQL-E-NAMTOOBIG error.

In addition, if a SET CONNECT or DISCONNECT statement were submitted with a name larger than 31 octets, the error text associated with the returned %SQL-E-NOSUCHCON error sometimes contained non-printing characters. This has been fixed so that these extraneous characters will no longer appear.

As a workaround, use a connection name of 31 octets or less.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.2.16 Loss of NULL Setting for Imported LIST OF BYTE VARYING Columns

Bug 4384906

In prior versions of Oracle Rdb V7.1, the IMPORT command would lose the NULL attribute for LIST OF BYTE VARYING columns. Instead these columns would be imported as empty lists.

The following example shows the difference before and after an IMPORT.

```
SQL> attach 'filename PERSONNEL';
SQL> insert into resumes values ('00167', null);
1 row inserted
SQL> select * from resumes;
  EMPLOYEE_ID  RESUME
  00167        NULL
1 row selected
SQL> commit;
SQL> export database alias RDB$DBHANDLE into PERS.RBR;
SQL> disconnect all;
SQL> drop database filename PERSONNEL;
SQL> import database from PERS.RBR filename PERSONNEL.RDB;
SQL> select * from resumes;
  EMPLOYEE_ID  RESUME
  00167        0:0:0
1 row selected
SQL> commit;
```

This problem affects all LIST OF BYTE varying columns in user defined tables.

## Oracle® Rdb for OpenVMS

The column's NULL attribute is lost and the segmented string id appears as 0:0:0 when displayed by Interactive SQL. If a cursor is opened on such LIST OF BYTE VARYING columns then it will act like an empty list and return no data, just as it would if the NULL attribute was on. However, applications that test a LIST column using IS NULL or IS NOT NULL will possibly match a different set of rows.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.3 RDO and RDML Errors Fixed

### 4.3.1 Unexpected NOT\_LARDY Following LOCK\_CONFLICT Exception

Bug 2274845

In prior releases of Oracle Rdb, when using the RDO or RDBPRE interfaces, it was possible to receive a RDMS-F-NOT\_LARDY after a RDB-F-LOCK\_CONFLICT occurred. This error would repeat for all access to the table which caused the LOCK\_CONFLICT error. A ROLLBACK or COMMIT was required to clear this error state.

The following example shows this behavior.

```
INVOKE DATABASE FILENAME TEST_DATABAS

START_TRANSACTION READ_WRITE CONCURRENCY NOWAIT

FOR CLI IN CLI_DNI WITH CLI.DNI_CLI = 'DNI37'
  PRINT CLI.COD_IDI,CLI.FE_ULT_CON
  MODIFY CLI USING
    CLI.COD_IDI='CAS'
  END_MODIFY
END_FOR
%RDB-E-LOCK_CONFLICT, request failed due to locked resource
-RDMS-F-LCKCNFLCT, lock conflict on logical area 57

FOR CLI IN CLI_DNI WITH CLI.DNI_CLI = 'DNI37'
  PRINT CLI.COD_IDI,CLI.FE_ULT_CON
  MODIFY CLI
    USING CLI.COD_IDI='CAS'
  END_MODIFY
END_FOR
%RDMS-F-NOT_LARDY, area for 57:1257:4 not in proper ready mode
```

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.4 RMU Errors Fixed

### 4.4.1 RMU Extract Might Extract Incomplete Routine Definition

Bug 4142629

In Oracle Rdb Release 7.1.3 and 7.1.4, RMU Extract could possibly extract stored procedures and functions incorrectly. This occurred when the original definition included a quoted string (such as in the COMMENT IS clause) that started in the first column of the source. The result was that the first statement in the routine would not be extracted.

A workaround is to start all string literals in the routine header after the first column.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

### 4.4.2 Unexpected ACCVIO and BUGCHECK from RMU Extract

Bug 4205822

In prior versions of Oracle Rdb, attempts to use RMU Extract to format a view, trigger or constraint definition that contained a dbkey literal (for example `_DBKEY'-1:-1:-1'`) would fail with errors similar to those shown below.

```
$ rmu/extract/item=view/option=(filename_only,noheader,match=myv%) sql$database
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename MF_PERSONNEL';
create view MYV
  (JRN,
   FMUB) as
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000000000000, PC=FFFFFFFF81096B64, PS=0000001B
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTER]RMUEXTBUGCHK.DMP;
%RMU-F-FTL_RMU, Fatal error for RMU operation at 26-FEB-2005 03:31:09.78
```

This problem has been corrected in Oracle Rdb Release 7.1.4.1. The following example shows the corrected output from RMU Extract. Please note that SQL converts `_ROWID` into `_DBKEY` literals so that this is what will be extracted by RMU.

```
$ rmu/extract/item=view/option=(filename_only,noheader,match=myv%) sql$database
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
```

```

attach 'filename MF_PERSONNEL';
create view MYV
  (JRN,
   FMUB) as
  (select
     case C1.DBKEY
       when _DBKEY'-1:-1:-1' then 'D'
       else 'I'
     end,
     C1.MY_UB
  from MYONE C1);

commit work;

```

### 4.4.3 RMU Load Did Not Handle UNSPECIFIED Character Set in RRD File

Bug 4242736

In prior releases of Oracle Rdb, the special character set UNSPECIFIED was not correctly supported by RMU Load. This character set informs Rdb that this data can be assigned to and from a character string of any other character set. It is the default character set for the USER, CURRENT\_USER, SESSION\_USER and SYSTEM\_USER builtin functions to allow assignment in any database.

The following example shows the error previously reported by RMU when it checks for compatible character sets between the source data file and the target column.

```

$ RMU /LOAD/RECORD_DEFINITION=(FILE=1.RRD,FORMAT=DELIMITED_TEXT,NULL="*") -
/TRANSACTION_TYPE=EXCLUSIVE/BUFFERS=400 MF_PERSONNEL DUMMY_COPY DUMMY.UNL
  DEFINE FIELD F1 DATATYPE IS TEXT SIZE IS 31 CHARACTERS CHARACTER SET IS
  UNSPECIFIED.
  DEFINE RECORD DUMMY.
  F1 .
  END DUMMY RECORD.
%RMU-F-FLDMUSMAT, Specified fields must match in number and datatype with the
unloaded data
%RMU-I-DATRECSTO,    0 data records stored.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 16-MAR-2005 14:54:20.21

```

This problem has been corrected in Oracle Rdb Release 7.1.4.1. This problem has been corrected so that RMU Load allows assignment from this data to any character column.

### 4.4.4 RMU/CONVERT Ignored Storage Maps Defined for Optional System Tables

Bug 4274119

RMU/CONVERT did not check to see if storage maps were defined for optional system tables. This caused an access violation since an invalid logical area ID of zero was retrieved from the system table RDB\$RELATIONS record for the optional table when the correct logical area ID was defined in the storage map. Also, it was assumed that optional system tables were stored compressed when the storage map could specify that compression was disabled. In addition, optional system tables with storage maps were not

converted if the record format had changed and therefore were not put in a new logical area by RMU/CONVERT. These problems have been fixed. Note that these problems only happened for optional system tables that used storage maps.

The following example shows that an access violation occurred because a storage map was defined for the optional system table RDB\$WORKLOAD. This caused RMU/CONVERT to use an invalid logical area ID of zero.

```

RYEROX>RMU/CONVERT TEST_DB.RDB
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.1-401
Are you satisfied with your backup of DEVICE:[DIRECTORY]TEST_DB.RDB;1
and your backup of any associated .aij files [N]? y
%RMU-I-LOGCONVRT, database root converted to current structure level
%RMU-S-CVTDBSUC, database DEVICE:[DIRECTORY]TEST_DB.RDB;1
successfully converted from version V7.0 to V7.1
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=
0000000000000168, PC=00000000003D5DA0, PS=0000001B
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-I-BUGCHKDMP, generating bugcheck dump file
DEVICE:[DIRECTORY]RMUBUGCHK.DMP;
%RMU-F-FTL_CNV, Fatal error for CONVERT operation at 22-APR-2005 10:11:57.83

```

As a workaround, do not define storage maps for optional system tables.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.4.5 RMU /UNLOAD From Remote Database Specification

Previously, the RMU /UNLOAD command to unload records from a database table did not allow the database specification to include a remote file specification. A RMU-F-NONODE fatal error would be returned when attempting to use a remote database specification, as in the following example.

```

$ DEFINE DB$ REMNOD::DUA0:[DB]DB.RDB
$ RMU/UNLOAD DB$ C1 C1
%RMU-W-BADDBNAME, can't find database root REMNOD::DUA0:[DB]DB.RDB;1
-RMU-F-NONODE, no node name is allowed in the file specification
%RMU-F-CANTOPNROO, cannot open root file "REMNOD::DUA0:[DB]DB.RDB;1"
%RMU-F-FTL_UNL, Fatal error for UNLOAD operation at 13-MAY-2005 02:18:26.61

```

This problem has been corrected in Oracle Rdb Release 7.1.4.1. The remote node operation is now allowed. Note, however, that the database root file may be accessed by both FAL and the Rdb database server (because not all of the RMU operations are strictly database user attaches).

## 4.4.6 RMU/SHOW AFTER/BACKUP May Stall When AIJs Are Full

Bug 4347617

In previous releases of Oracle Rdb, when using multiple circular journals which have a state of Full, the RMU/SHOW AFTER/BACKUP command may stall on "waiting for AIJ journal lock 0 (PW)".

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.4.7 Problem with LITERAL Character Set and Embedded Quotes in RMU EXTRACT

Bug 4469731

In prior releases of Oracle Rdb, it was possible that RMU/EXTRACT might generate incorrect code for the session LITERAL character set definition if the database had a default character set other than DEC\_MCS.

```
$ rmu/extract/item=(tables) test/out=test.sql
$ sql @test.sql;
SQL> set language ENGLISH;
SQL> set default date format 'SQL92';
SQL> set quoting rules 'SQL92';
SQL> set date format DATE 001, TIME 001;
SQL> attach 'filename device:[directory]TEST.RDB';
SQL> set default character set 'DEC_HANYU';
SQL> set literals character set 'DEC_HANYU';
%SQL-I-SPELLCORR, identifier LITERALS replaced with LITERAL
```

In addition, if the database default character was not DEC\_MCS, embedded single quotation marks were not correctly doubled in object comments.

```
...
SQL> create table T (
cont>     F
cont>     INTEGER)
cont>     comment is
cont>     'This relation will be used in testing 'store' statments';
      'This relation will be used in testing 'store' statments';
      ^
%SQL-W-LOOK_FOR_STT, Syntax error, looking for:
%SQL-W-LOOK_FOR_CON,           /, ENABLE, STORAGE, LOGGING, COMMENT,
DISABLE, PCTFREE, PCTUSED, INITRANS, MAXTRANS,
%SQL-W-LOOK_FOR_CON,           NOLOGGING, TABLESPACE, ;,
%SQL-F-LOOK_FOR_FIN,         found STORE instead
SQL>
```

These problems have been corrected in Oracle Rdb Release 7.1.4.1.

## 4.5 LogMiner Errors Fixed

### 4.5.1 Incorrect Elimination of AIJ When Using RMU /UNLOAD /AFTER\_JOURNAL /ORDER\_AIJ\_FILES /RESTART

Bug 4121598

In previous versions of Oracle Rdb, it was possible for the specified input after-image journal file to be skipped when using the /RESTART and /ORDER\_AIJ\_FILES qualifiers and only a single input AIJ file. The single after-image journal file would be incorrectly eliminated from processing in this case.

This problem has been corrected in Oracle Rdb Release 7.1.4.1. The after-image journal file will not be eliminated when using the /RESTART and /ORDER\_AIJ\_FILES qualifiers and only a single input AIJ file.

### 4.5.2 RMU /UNLOAD /AFTER\_JOURNAL /[NO]SYMBOLS

Previously, it was possible for a large number of DCL symbols to be created by the RMU /UNLOAD /AFTER\_JOURNAL command. If enough tables were being unloaded, the CLI symbol table space could become exhausted. The error message "LIB-F-INSCLIMEM, insufficient CLI memory" would be returned in this case.

As a possible workaround, increasing the system parameter CLISYMTBL will provide more space for the CLI symbol table.

This problem has been corrected in Oracle Rdb Release 7.1.4.1. A qualifier "/[NO]SYMBOLS" has been added to the RMU /UNLOAD /AFTER\_JOURNAL command. The default is "/SYMBOLS" to cause the symbols to be created. Specifying "/NOSYMBOLS" prevents the symbols being created.

### 4.5.3 RMU /UNLOAD /AFTER\_JOURNAL Incorrect Settings in Null Bit Vector

With some combinations of table definition modifications with adding and removing fields, it was possible for a table's maximum null bit vector length to be incorrectly calculated by the Oracle Rdb LogMiner(tm). The incorrect length would be used internally by the RMU /UNLOAD /AFTER\_JOURNAL command and could result in an incorrect null bit vector content being generated for the output stream.

As a possible workaround, unloading the table data from the database, dropping and recreating the table and then reloading the content would cause the table's maximum field identification to be reset and the RMU /UNLOAD /AFTER\_JOURNAL command would then work correctly.

This problem has been corrected in Oracle Rdb Release 7.1.4.1. The RMU /UNLOAD /AFTER\_JOURNAL command now correctly determines the maximum field ID and the null bit vector length.

## 4.5.4 RMU /UNLOAD /AFTER\_JOURNAL Field Order Clarification

Unlike SQL and RMU /UNLOAD, the Oracle Rdb LogMiner (tm) RMU /UNLOAD /AFTER\_JOURNAL command outputs fields in the database on-disk field order. This difference can become apparent when a table definition has been modified. For example, when adding a new column to appear in a position other than at the end of the record.

The following example shows one way that the Oracle Rdb LogMiner (tm) RMU /UNLOAD /AFTER\_JOURNAL command outputs fields in the database on-disk field order.

```

$!
$ SQL$
  CREATE DATA FILE FOO LOGMINER SUPPORT ENA;

  -- Create table then insert another column between COL1 & COL2.

  CREATE TABLE T1 (COL1 INT, COL2 INT);
  ALTER TABLE T1 ADD COLUMN COL3 INT BEFORE COLUMN COL2;
  SHOW TABLE (COLUMN) T1;
Information for table T1

Columns for table T1:
Column Name                Data Type                Domain
-----
COL1                        INTEGER
COL3                        INTEGER
COL2                        INTEGER

  COMMIT;

  DISCONNECT ALL;
  ALTER DATA FILE FOO JOU ENA ADD JOU J1 FILE J1;
  EXIT;
$!
$ RMU/BACKUP/NOLOG FOO NLA0:FOO
$ RMU/BACKUP/AFTER/NOLOG FOO NLA0:B1
$!
$ SQL$
  ATTACH 'FILE FOO';
  INSERT INTO T1 VALUES (NULL,2,3); -- COL1=NULL, COL3=2, COL2=3
1 row inserted
  INSERT INTO T1 VALUES (1,NULL,3); -- COL1=1, COL3=NULL, COL2=3
1 row inserted
  INSERT INTO T1 VALUES (1,2,NULL); -- COL1=1, COL3=2, COL2=NULL
1 row inserted
  COMMIT;
  SELECT * FROM T1; -- Show data content
      COL1          COL3          COL2
      NULL          2          3
      1             NULL         3
      1             2          NULL
3 rows selected
  EXIT;
$!
$ RMU/BACKUP/AFTER/NOLOG FOO B1
$!
$ RMU/UNL/RECORD=FILE=T1.RRD1 FOO T1 NLA0:T1

```

## Oracle® Rdb for OpenVMS

```
%RMU-I-DATRECUNL, 3 data records unloaded.
$ RMU/UNL/AFTER/NOLOG/FORMAT=DUMP FOO B1 -
  /TABLE=(NAME=T1,OUTPUT=T1.DAT,RECORD=T1.RRD2)
$!
$ SEARCH T1.RRD1 COL ! Show field order - RMU/UNLOAD
DEFINE FIELD COL1 DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD COL3 DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD COL2 DATATYPE IS SIGNED LONGWORD.
  COL1 .
  COL3 .
  COL2 .
$ SEARCH T1.RRD2 COL ! Show field order - RMU/UNLOAD/AFTER_JOURNAL
DEFINE FIELD COL1 DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD COL2 DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD COL3 DATATYPE IS SIGNED LONGWORD.
  COL1.
  COL2.
  COL3.
$ SEARCH T1.DAT RDB$LM_RELATION_NAME, COL ! Show data content
RDB$LM_RELATION_NAME      : T1
COL1                      : NULL
COL2                      : 3
COL3                      : 2
RDB$LM_RELATION_NAME      : T1
COL1                      : 1
COL2                      : 3
COL3                      : NULL
RDB$LM_RELATION_NAME      : T1
COL1                      : 1
COL2                      : NULL
COL3                      : 2
$!
```

## 4.6 Row Cache Errors Fixed

### 4.6.1 RMU /CLOSE /ABORT=DELPRC /WAIT Hang

Bug 4304166

Starting with Oracle Rdb V7.1, it is possible that when using the Row Cache feature, closing a database with the /ABORT=DELPRC and /WAIT qualifiers may hang. The problem results in the database recovery processes and the record cache server process becoming deadlocked. To actually close the database may require manual intervention to explicitly STOP/ID the database recovery processes.

With this combination of command line qualifiers, the Oracle Rdb monitor process was incorrectly issuing a \$DELPRC to the record cache server process. The /ABORT=DELPRC and /ABORT=FORCEX qualifiers are intended to apply only to database user processes and not to database servers when /WAIT is not specified.

This problem has been corrected in Oracle Rdb Release 7.1.4.1. The Oracle Rdb monitor process no longer attempts to issue a \$DELPRC to the record cache server process when /WAIT is specified.

### 4.6.2 Long Running Transaction Hangs After RMU/CHECKPOINT

Bug 4290880

When row caching was enabled, after an RMU/CHECKPOINT command was issued, it was possible for long running update transactions to hang waiting for the global checkpoint lock. The RMU/SHOW STATISTICS "Stall Messages" screen would show output similar to the following:

```
Process.ID Since..... T Stall.reason.....Lock.ID.
2360128C:2 05:42:54.38 W waiting for global checkpoint (CR)      1B00783D
23601292:1u 05:42:54.39 - waiting for RCS synch. request 1:23601291 (EX)
23601291:1s 05:42:54.72 - waiting for record 97:12875:0 (PR)      5600322C
23601291:1s 05:42:54.72 - waiting for record 97:12875:0 (PR)      5600322C
```

This problem would occur because under the old checkpointing mechanisms a process had to obtain the global checkpoint lock prior to making any further updates to the database. Thus a long running update transaction would stall waiting for the RMU process to release the global checkpoint lock. The RMU process would not release the global checkpoint lock until the Row Cache Server (RCS) process completed its checkpoint. The RCS could not complete its checkpoint until the long running transaction released its row locks. Thus all processes would stall in a deadlock.

This problem has been resolved by changing the checkpoint mechanism to use an asynchronous lock request to obtain the global checkpoint lock. That is, processes no longer stall waiting to obtain the lock but instead queue a lock request and then continue processing. This prevents processes from stalling for the global checkpoint lock.

This problem can be avoided by not using the RMU/CHECKPOINT command. If a checkpoint timeout interval is declared for the database, then forcing checkpoints should not be necessary. The "CHECKPOINT TIMED EVERY <n> SECONDS" clause can be used to enable checkpoint timers. This problem can also be

avoided by disabling row cache.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

## 4.7 Hot Standby Errors Fixed

### 4.7.1 Excessive CPU Consumed by LRS Process

Bug 4268610

When using the Hot Standby feature, the AIJ Log Recovery Server (LRS) process would sometimes consume inordinate amounts of CPU. This would especially be true when the following conditions were true:

- Many short duration transactions were being generated by the application.
- Many journal slots were allocated in the database.
- The current journal sequence number was relatively large, for example, in the tens of thousands.

When the above conditions were true, it exposed a problem in the recovery code statistics collection. The recovery code was attempting to determine how many blocks of journal were spanned in each committed transaction. However, the recovery code did not record the journal starting point for the transaction, thus it would appear that the transaction started in journal sequence number 0. The statistics code would then first attempt to find journal sequence 0. When it didn't find sequence 0 it would then try to find sequence 1 so that it could approximate the number of blocks in the transaction. That journal would not be found so it would continue searching for journals until it tried all possible journal sequence numbers up to the current journal. This could consume huge amounts of CPU time.

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.4.1. The recovery code will no longer attempt to gather statistics on the number of journal blocks per transaction since the number is not useful in the context of database recovery.

### 4.7.2 LRS Bugchecks in DIOLAREA\$SCAN\_ABM\_CHAIN

Bug 4429420

The Hot Standby Log Recovery Server (LRS) process could fail with a bugcheck exception similar to the following when replication is first started for a restored standby database.

```
***** Exception at 00225AC4 : DIOLAREA$SCAN_ABM_CHAIN + 000003E4
%COSI-F-BUGCHECK, internal consistency failure
```

This problem would occur when the current journals for the master database contained entries for a DROP TABLE or DROP INDEX operation, but the database backup used to create the standby database did not contain the table or index that was dropped. That is, the database backup was done after the table or index were dropped.

This problem can be demonstrated with the following sequence of commands:

```
$ ! Create master database.
$
$ SQL$
CREATE DATABASE FILENAME TEST_MASTER
```

## Oracle® Rdb for OpenVMS

```
CREATE STORAGE AREA RDB$SYSTEM FILENAME TEST_MASTER;
DISCONNECT ALL;
ALTER DATABASE FILENAME TEST_MASTER RESERVE 2 JOURNALS
OPEN IS MANUAL;
%RDMS-W-DOFULLBCK, full database backup should be done to ensure future recovery
ALTER DATABASE FILENAME TEST_MASTER JOURNAL IS ENABLED
(FAST COMMIT IS ENABLED,
LOG SERVER IS AUTOMATIC)
ADD JOURNAL AIJ_1 FILENAME 'SYS$DISK:[ ]TEST_MASTER_JOURNAL1'
ADD JOURNAL AIJ_2 FILENAME 'SYS$DISK:[ ]TEST_MASTER_JOURNAL2'
ADD JOURNAL AIJ_3 FILENAME 'SYS$DISK:[ ]TEST_MASTER_JOURNAL3';
%RDMS-W-DOFULLBCK, full database backup should be done to ensure future recovery
EXIT;
$
$ RMU/OPEN TEST_MASTER
$
$ ! Create a table
$
$ SQL$
ATTACH 'FILENAME TEST_MASTER';
CREATE TABLE T1 (C1 INT);
INSERT INTO T1 VALUES (1);
1 row inserted
COMMIT;
EXIT;
$
$ ! Dispose of the journal entries reflecting the CREATE TABLE.
$
$ RMU/BACKUP/AFTER/NOLOG TEST_MASTER NL:
%RMU-W-DATACMIT, unjournalized changes made; database may not be recoverable
$
$ ! Drop the table. The journal will contain the DROP TABLE but will not
$ ! contain the original CREATE TABLE.
$
$ SQL$
ATTACH 'FILENAME TEST_MASTER';
DROP TABLE T1;
COMMIT;
EXIT;
$
$ ! Now that the table is gone, backup the database and use the backup to
$ ! create the standby database. The standby database will not contain an
$ ! ABM page for the table that was CREATED/DROPPed, but the journal will
$ ! contain the DROP TABLE entries. The LRS will process the DROP TABLE
$ ! journal entries for the non-existent table.
$
$ RMU/BACKUP/NOLOG/ONLINE TEST_MASTER TEST_MASTER
$ RMU/SHOW AFTER_JOURNAL/OUTPUT=TEST_AIJ.OPT TEST_MASTER
$ RMU/RESTORE/NOLOG/NOCD/NEW/DIR=SYS$DISK:[ ]-
  /ROOT=SYS$DISK:[ ]TEST_STANDBY -
  /AIJ_OPT=TEST_AIJ.OPT TEST_MASTER
JOURNAL IS ENABLED -
RESERVE 3 -
ALLOCATION IS 512 -
EXTENT IS 512 -
OVERWRITE IS DISABLED -
SHUTDOWN_TIMEOUT IS 60 -
NOTIFY IS DISABLED -
BACKUPS ARE MANUAL -
CACHE IS DISABLED
ADD JOURNAL AIJ_1 -
! FILE SYS$DISK:[ ]TEST_MASTER_JOURNAL1.AIJ;1
```

## Oracle® Rdb for OpenVMS

```
FILE SYS$DISK:[ ]TEST_MASTER_JOURNAL1
ADD JOURNAL AIJ_2 -
! FILE SYS$DISK:[ ]TEST_MASTER_JOURNAL2.AIJ;1
FILE SYS$DISK:[ ]TEST_MASTER_JOURNAL2
ADD JOURNAL AIJ_3 -
! FILE SYS$DISK:[ ]TEST_MASTER_JOURNAL3.AIJ;1
FILE SYS$DISK:[ ]TEST_MASTER_JOURNAL3
$ RMU/OPEN TEST_STANDBY
$ RMU/REPLICATE AFTER_JOURNAL START TEST_STANDBY -
  /MASTER_ROOT=TEST_MASTER -
  /OUTPUT=TEST_LRS.LOG
%RMU-I-HOTOFFLINE, standby database opened for exclusive access
$
$ ! Start replication. The LRS will attempt to apply the REL_CLUMP entry.
$ ! Part of processing the REL_CLUMP is to update the ABM pages, but they
$ ! don't exist. As reported in BUG 4429420, the LRS would bugcheck at
$ ! DIOLAREA$SCAN_ABM_CHAIN + 000003E4. In this test the following REPLICATE
$ ! START command would return an RDMS-F-TIMEOUT error when the LRS crashed.
$ ! The LCS timed out waiting for the LRS to respond.
$
$ RMU/REPLICATE AFTER_JOURNAL START TEST_MASTER /STANDBY_ROOT=TEST_STANDBY
%RDMS-F-TIMEOUT, timeout on
```

This problem can be avoided by backing up the journals and the database in sequence without doing any intervening metadata operations.

This problem has been corrected in Oracle Rdb Release 7.1.4.1.

---

# **Chapter 5**

## **Enhancements Provided in Oracle Rdb Release 7.1.4.2**

# 5.1 Enhancements Provided in Oracle Rdb Release 7.1.4.2

## 5.1.1 /TRANSPORT Added to RMU/REPLICATE AFTER

Bug 4109344

The /TRANSPORT qualifier has been added to the RMU/REPLICATE AFTER START and CONFIGURE commands. This new qualifier allows the network transport to be specified. The valid values are "DECNET" and "TCPIP". The specified network transport is saved in the database.

In previous releases, to use TCP/IP as the network transport for Hot Standby, the system-wide logical "RDM\$BIND\_HOT\_NETWORK\_TRANSPORT" had to be defined.

See the following example of this new feature.

```
$ RMU/REPLICATE AFTER CONFIGURE /TRANSPORT=TCPIP  
/STANDBY=REMNOD::DEV:[DIR]STANDBY_DB M_TESTDB
```

## 5.1.2 CASCADE Option Now Supported by DROP INDEX

In previous versions of Oracle Rdb, the option CASCADE was not supported. This meant that a DROP INDEX of an index used in the PLACEMENT VIA INDEX clause in a storage map would fail. The following example shows the generated error when the EMPLOYEES\_HASH is dropped from the EMPLOYEES table in the MF\_PERSONNEL database.

```
SQL> drop index EMPLOYEES_HASH;  
%RDB-E-NO_META_UPDATE, metadata update failed  
-RDMS-F-INDINMAP, index "EMPLOYEES_HASH" is used in storage map "EMPLOYEES_MAP"  
SQL> drop index EMPLOYEES_HASH cascade;  
SQL>
```

### Syntax



### Arguments

- CASCADE  
Specifies that you want SQL to modify any storage map that uses this index to be a NO PLACEMENT VIA INDEX storage map.

- **IF EXISTS**  
Prevents SQL command language from displaying error messages if the referenced index does not exist in the database.
- **index-name**  
Identifies the name of the index you wish to delete.
- **RESTRICT**  
Prevents the removal of an index if it is referenced by any other object within an Oracle Rdb database. RESTRICT is the default.

### *Usage Notes*

- CASCADE will implicitly alter the storage map and change it to a NO PLACEMENT VIA INDEX storage map.
- Any query outline that references the index being dropped will be marked invalid for both RESTRICT and CASCADE options of the DROP INDEX command.
- In a multischema database, the DROP INDEX ... CASCADE statement will be used implicitly to support the DROP SCHEMA ... CASCADE statement. In previous versions of Oracle Rdb, this statement would fail if a storage map referenced an index that was to be dropped.

## **5.1.3 Oracle Rdb Release 7.1.x.x New Features Document Added**

A new document has been created which contains all of the New Features Chapters from all previous Rdb 7.1 Release Notes. This document will be included in saveset A of the Rdb kit. It is called NEWFEATURES\_71xx and will be available in postscript, text and PDF format. This will provide customers with one document to reference to find out about all new features that have been added to the Rdb 7.1 releases.

## **5.1.4 READER\_THREAD\_RATIO and RMU BACKUP**

Prior to Oracle Rdb V7.1, RMU BACKUP created one reader thread to read each storage area in the storage area list that was assigned to a writer thread. This could cause a decrease in performance due to reader thread overhead caused by a larger number of reader threads competing for system resources, especially if a large number of storage areas was assigned to a writer thread.

Starting with Oracle Rdb Release 7.1.0, a reader thread pool has been used by default with the RMU BACKUP of a database to enhance performance by lessening the reader thread overhead caused by having a larger number of reader threads executing at the same time.

By default, 5 reader threads are created and assigned to each writer thread to read the storage areas assigned to that thread for backup. If fewer than 5 areas are assigned to a writer thread, one reader thread is created to read each storage area.

If the user does not want to use the default of 5 reader threads for each writer thread on a database backup because this default does not give the best backup performance for a particular database the

```
/READER_THREAD_RATIO = n
```

qualifier has been added where "n" specifies the number of reader threads to be created and assigned to each writer thread. The default is 5 or the actual number of storage areas assigned to the writer thread if it is less than 5. If "0" is specified, a thread pool is not used. The pre V7.1 method is used where the number of reader threads created and assigned to a writer thread equals the number of storage areas that writer thread is writing to the backup media. Therefore, the minimum value for using a thread pool is 1. There is no specific maximum number. If the specified maximum number of reader threads exceeds the number of storage areas assigned to a writer thread, it will be reduced to equal that number of storage areas. Therefore, no reader threads will be created without work to do.

The following example shows /READER\_THREAD\_RATIO used with a database backup to specify that 2 reader threads are to be used with each writer thread instead of the default of 5.

```
$ RMU/BACKUP/NOLOG/READER_THREAD_RATIO=2 MF_PERSONNEL MFP.RBF
$ RMU/RESTORE/NEW/NOCD MFP.RBF
%RMU-I-AIJRSTAVL, 0 after-image journals available for use
%RMU-I-AIJISOFF, after-image journaling has been disabled
%RMU-W-USERECCOM, Use the RMU Recover command. The journals are not available.
```

The following example shows /READER\_THREAD\_RATIO=0 used with a database backup to specify that one reader thread should be created for each storage area assigned to a writer thread and not a maximum of 5 (in other words, use the pre V7.1 behavior).

```
$ RMU/BACKUP/NOLOG/READER_THREAD_RATIO=0 MF_PERSONNEL MFP.RBF
$ RMU/RESTORE/NEW/NOCD MFP.RBF
%RMU-I-AIJRSTAVL, 0 after-image journals available for use
%RMU-I-AIJISOFF, after-image journaling has been disabled
%RMU-W-USERECCOM, Use the RMU Recover command. The journals are not available.
```

## 5.1.5 RMU/DUMP/AFTER\_JOURNAL AREA, PAGE and LINE QUALIFIERS

Three useful qualifiers that are currently undocumented but will be put in the Oracle Rdb RMU V7.2 Reference Manual and in Oracle Rdb RMU V7.1 and V7.2 online help can be used with the *RMU/DUMP/AFTER\_JOURNAL name.aj* command which dumps the content of a database after image journal file. These qualifiers can be used to limit the information to be dumped.

```
/AREA=#
```

```
# = database storage area number
minimum value is 1
```

```
/PAGE=#
```

```
# = database page number
minimum value is 1
```

```
/LINE=#
```

```
# = database line number
minimum value is 0
```

All three qualifiers cannot be negated and require a numeric value.

## Oracle® Rdb for OpenVMS

These qualifiers can be used in the dump of an Rdb after image journal file to only dump information from the journal file related to the specified database area and/or page and/or line.

The following example dumps the information from a database after image journal file related to the specified area, line and page.

```
$rmu/dump/after_journal/area=3/page=560/line=1 mf_personnel.aij
```

```
*-----  
* Oracle Rdb X7.1-00                               3-NOV-2005 10:42:23.56  
*  
* Dump of After Image Journal  
*   Filename: DEVICE:[DIRECTORY]MF_PERSONNEL.AIJ;1  
*  
*-----
```

```
2/4          TYPE=D, LENGTH=122, TAD= 3-NOV-2005 10:31:12.56, CSM=00  
TID=6, TSN=0:640, AIJBL_START_FLG=01, FLUSH=00, SEQUENCE=1  
MODIFY: PDBK=3:560:1, LDBID=0, PSN=0, FLAGS=00, LENGTH=84
```

```
          0022 0000 line 1 (3:560:1) record type 34  
          00 0001 0002 Control information  
          .... 79 bytes of static data  
86726576696C6F54343631303000010D 0005 data '...00164Toliver.'  
5020363431411120846E69766C410420 0015 data ' .Alvin. .A146 P'  
009820876563616C50206C6C656E7261 0025 data 'arnell Place. ...'  
3330484E12208B6175726F636F684307 0035 data '.Chocorua. .NH03'  
20F03100630F72B31C00004D373138 0045 data '817M...³r.c.1'
```

```
2/6          TYPE=D, LENGTH=224, TAD= 3-NOV-2005 10:31:12.56, CSM=00  
TID=6, TSN=0:641, AIJBL_START_FLG=01, FLUSH=00, SEQUENCE=3  
MODIFY: PDBK=3:560:1, LDBID=0, PSN=1, FLAGS=00, LENGTH=84
```

```
          0022 0000 line 1 (3:560:1) record type 34  
          00 0001 0002 Control information  
          .... 79 bytes of static data  
86726576696C6F54343631303000010D 0005 data '...00164Toliver.'  
5020363431411120846E69766C410420 0015 data ' .Alvin. .A146 P'  
009820876563616C50206C6C656E7261 0025 data 'arnell Place. ...'  
3330484E12208B6175726F636F684307 0035 data '.Chocorua. .NH03'  
20F03100630F72B31C00004D373138 0045 data '817M...³r.c.1'
```

```
3/9          TYPE=D, LENGTH=330, TAD= 3-NOV-2005 10:31:12.73, CSM=00  
TID=6, TSN=0:642, AIJBL_START_FLG=01, FLUSH=00, SEQUENCE=5  
MODIFY: PDBK=3:560:1, LDBID=0, PSN=2, FLAGS=00, LENGTH=84
```

```
          0022 0000 line 1 (3:560:1) record type 34  
          00 0001 0002 Control information  
          .... 79 bytes of static data  
86726576696C6F54343631303000010D 0005 data '...00164Toliver.'  
5020363431411120846E69766C410420 0015 data ' .Alvin. .A146 P'  
009820876563616C50206C6C656E7261 0025 data 'arnell Place. ...'  
3330484E12208B6175726F636F684307 0035 data '.Chocorua. .NH03'  
20F03100630F72B31C00004D373138 0045 data '817M...³r.c.1'
```

Use "/AIJ\_BUFFERS=3" when recovering this AIJ journal. Make sure you have enough working set and pagefile quota for the recommended number of buffers.

## Oracle® Rdb for OpenVMS

1 recovery buffer referenced 3 times (3:559-561): 50.0%  
1 recovery buffer referenced 2 times (3:436-438): 33.3%  
1 recovery buffer referenced 1 time (3:134-136): 16.6%

Journal effectiveness: 54.5%

3 data records  
6 data modification records  
11 total modification records  
3 commit records

---

# **Chapter 6**

## **Enhancements Provided in Oracle Rdb Release 7.1.4.1**

# 6.1 Enhancements Provided in Oracle Rdb Release 7.1.4.1

## 6.1.1 Support Added for ANSI C Comments

For several years, the ANSI C standard has included the `"/"` style comment in addition to the traditional `"/*`  
`*/` style comment. Until now, SQL\$PRE/CC has only supported the later style. Support for the `"/"` style comment has been added.

The following example shows comments in the old and in the newly supported style.

```
/*
 * Traditional C style comments
 */
i0 += 1;          /* add one to i0 */

//
// Comments using the // style
//
i0 += 1;          // add one to i0
```

## 6.1.2 New Rdb Character Set GB18030

Oracle Rdb has introduced a new Chinese character set, GB18030, which is defined by the GB18030–2000 standard as used by the People's Republic of China. GB18030 encodes characters in sequences of one, two, or four octets. The following are valid octet sequences:

```
§      Single-octet: %X'00'–%X'7F'

§      Two-octet: %X'81'–%X'FE' + %X'40'–%X'7E' %X'80'–%X'FE';

§      Four-octet: %X'81'–%X'FE' + %X'30'–%X'39' %X'81'–%X'FE' + %X'30'–%X'39'
```

The single–octet characters comply with the standard GB 11383 (iISO 4873:1986). GB18030 has 1.6 million valid octet sequences.

As GB18030 contains single–octet ASCII characters, it may be used as an Identifier Character Set.

## 6.1.3 New GET DIAGNOSTICS Keyword

The following new keyword has been added to GET DIAGNOSTICS:

- **TRACE\_ENABLED**  
Returns an INTEGER value to indicate if the TRACE flag has been enabled using the statement SET FLAGS 'TRACE' or by either of the logical names RDMS\$SET\_FLAGS or RDMS\$DEBUG\_FLAGS. A zero (0) is returned if the flag is disabled, otherwise a one (1) is returned to indicate that tracing is enabled.

The following example shows usage of this keyword in a compound statement.

```
SQL> declare :x integer;
SQL> begin
cont> get diagnostics :x = TRACE_ENABLED;
cont> end;
SQL> print :x;
          X
          0
SQL> set flags 'trace';
SQL> begin
cont> get diagnostics :x = TRACE_ENABLED;
cont> end;
SQL> print :x;
          X
          1
```

## 6.1.4 Buffer Memory Now Exported/Imported

Bug 4213762

The Recovery Journal setting for Buffer Memory is now EXPORTed/IMPORTed.

An example of the syntax follows:

```
create data file rujmem;
export data file rujmem into rujmem;
drop data file rujmem;
import data from rujmem file rujmem recovery journal (buffer memory is local);
```

This feature has been added in Oracle Rdb Release 7.1.4.1.

## 6.1.5 RMU /UNLOAD Qualifier /REOPEN\_COUNT

Bug 4246050

Previously, RMU /UNLOAD would write a single output file for all records in the table/view being unloaded. In some cases, this single output file would become difficult to manipulate.

This problem has been corrected in Oracle Rdb Release 7.1.4.1. The "/REOPEN\_COUNT=n" qualifier allows specifying how many records will be written to an output file. The output file will be re-created (ie, a new version of the file created) when the record count reaches the specified value. The "/REOPEN\_COUNT=n" qualifier is only valid when used with the "/RECORD\_DEFINITION" or "/RMS\_RECORD\_DEF" qualifiers.

## 6.1.6 New DEFAULTS Qualifier Added to RMU Extract

This release of Oracle Rdb, Release 7.1.4.1, adds a new DEFAULTS qualifier to RMU Extract.

- Defaults=defaults-list

This qualifier is used to change the output of the RMU Extract command. The following defaults can be modified with the Defaults qualifier:

- ◆ Allocation
  - NoAllocation
  - When creating a test database using the RMU Extract generated script, the allocation from the source database may not be appropriate. The ALLOCATION keyword can be used to specify an alternate value to be used by all storage areas, or the NOALLOCATION keyword can be used to cause the clause to be omitted from the CREATE STORAGE MAP syntax. The default behavior, that is when neither keyword is used, is to use the allocation recorded in the database for each storage area. See also the SNAPSHOT\_ALLOCATION keyword.
- ◆ Date\_format
  - NoDate\_format
  - By default, RMU Extract assumes that DATE types will be SQL standard compliant (that is DATE ANSI) and that the builtin function CURRENT\_TIMESTAMP will return a TIMESTAMP(2) value. If your environment uses DATE VMS exclusively, then you can modify the default by specifying the default DATE\_FORMAT=VMS. The legal values are described in the Oracle Rdb SQL Reference Manual in the SET DEFAULT DATE FORMAT section. The default is DATE\_FORMAT=SQL92.
  - Use NODATE\_FORMAT to omit the setting of this session attribute from the script.
- ◆ Dialect
  - NoDialect
  - For some extracted SQL scripts, the language dialect is required to be specified. The DIALECT keyword can be used to supply a specified dialect for the script. The legal values for this option can be found in the Oracle Rdb SQL Reference Manual in the SET DIALECT section. The default is NODIALECT.
- ◆ Language
  - NoLanguage
  - RMU Extract uses the process language, that is the translated value of SYS\$LANGUAGE or ENGLISH, for the SET LANGUAGE command. However, if the script is used on a different system then this language might not be appropriate. The LANGUAGE keyword can be used to supply a specified language for the script. Legal language names are defined by the OpenVMS system logical name table. Examine the logical name SYS\$LANGUAGES for a current set. Use NOLANGUAGE to omit this command from the script.
- ◆ Quoting\_rules
  - NoQuoting\_rules
  - The QUOTING\_RULES keyword can be used to supply a specified setting for the script. The legal values for this option can be found in the Oracle Rdb SQL Reference Manual in the SET QUOTING RULES section. The default is QUOTING\_RULES=SQL92. Please note that RMU Extract assumes that SQL keywords and names containing non-ASCII character set values will be quoted.
- ◆ Snapshot\_allocation
  - NoSnapshot\_allocation
  - When creating a test database from the RMU Extract output, the snapshot file allocation from the source database may not be appropriate. The SNAPSHOT\_ALLOCATION keyword can be used to specify an alternate value to be used by all snapshot areas, or the NOALLOCATION keyword can be used to cause the "snapshot allocation is" clause to be omitted. The default behavior, that is when neither keyword is used, is to use the snapshot allocation stored in the database for each snapshot area. See also the ALLOCATION keyword.

## 6.1.7 New RESTART WITH Clause for ALTER SEQUENCE

This release of Oracle Rdb, 7.1.4.1, includes support for the ALTER SEQUENCE ... RESTART WITH clause. RESTART WITH allows the database administrator to reset the sequence to a specified value. The value must be within the range of MINVALUE and MAXVALUE. This command requires exclusive access to the sequence. Once the ALTER SEQUENCE statement is successfully committed, applications using the sequence will start with a value based on the restarted value.

---

### Note

***TRUNCATE TABLE for a table with an IDENTITY column implicitly executed an ALTER SEQUENCE ... RESTART WITH on the sequence. Specify the MINVALUE if it is an ascending sequence or MAXVALUE if it is a descending sequence.***

---

The following example shows the new RESTART WITH clause.

```
SQL> show sequence NEW_EMPLOYEE_ID
      NEW_EMPLOYEE_ID
Sequence Id: 1
Initial Value: 472
...
SQL>
SQL> alter sequence NEW_EMPLOYEE_ID
cont> restart with 500;
SQL>
SQL> show sequence NEW_EMPLOYEE_ID
      NEW_EMPLOYEE_ID
Sequence Id: 1
Initial Value: 500
...
SQL>
```

## 6.1.8 ALTER VIEW Statement

### *Description*

This statement allows the named view to be modified.

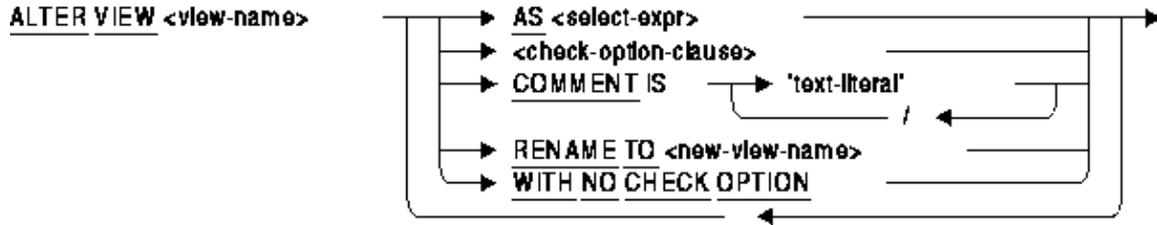
### *Environment*

You can use the ALTER VIEW statement:

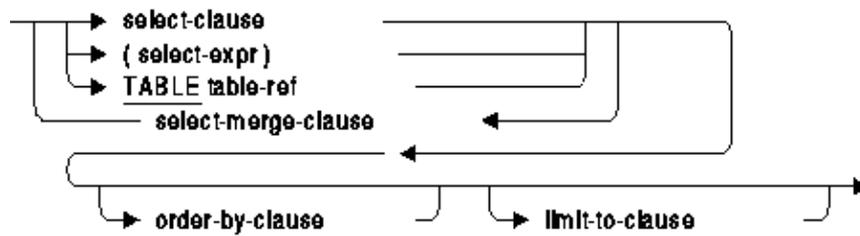
- In interactive SQL
- Embedded in host language programs
- As part of a procedure in a SQL module
- In dynamic SQL as a statement to be dynamically executed

### *Format*

---



**select-expr =**



**check-option-clause =**



**Arguments**

- **AS**  
Replaces the view select expression and the definitions of the columns. The number of expressions in the select list must match the original CREATE VIEW column list.
- **CONSTRAINT check-option-name**  
Specify a name for the WITH CHECK OPTION constraint. If you omit the name, SQL creates a name. However, Oracle Rdb recommends that you always name constraints. If you supply a name for the WITH CHECK OPTION constraint, the name must be unique in the schema.  
The name for the WITH CHECK OPTION constraint is used by the INTEG\_FAIL error message when an INSERT or UPDATE statement violates the constraint.
- **COMMENT IS**

Replaces the comment currently defined for the view (if any). The comment will be displayed by the SHOW VIEW statement in Interactive SQL.

- **RENAME TO**  
Renames the current view. The new view name must not exist as the name of an existing view, table, sequence or synonym.
- **WITH CHECK OPTION**  
A constraint that places restrictions on update operations made to a view. The check option clause ensures that any rows that are inserted or updated in a view conform to the definition of the view. Do not specify the WITH CHECK OPTION clause with views that are read-only. (The Usage Notes describe which views SQL considers read-only.)
- **WITH NO CHECK OPTION**  
Removes any check option constraint currently defined for the view.

### *Usage Notes*

- You require ALTER privilege on the referenced view.
- The ALTER VIEW statement causes the RDB\$LAST\_ALTERED column of the RDB\$RELATIONS table for the named view to be updated with the transaction's timestamp.
- Neither the column names nor their position may be modified using ALTER VIEW. Nor can columns be added or dropped for a view. These changes require both a DROP VIEW and CREATE VIEW statement to replace the existing definition.
- **RENAME TO** allows the name of the view to be changed. This clause requires that synonyms are enabled in the database. Use ALTER DATABASE ... SYNONYMS ARE ENABLED. The old name will be used to create a synonym for the new name of this view. This synonym can be dropped if the name is no longer used by database definitions or applications. This clause is equivalent to the RENAME VIEW statement.
- The **COMMENT IS** clause changes the existing comment on the view. This clause is equivalent to the COMMENT ON VIEW statement.
- Changes to the column expression may change the column to read-only and prevent referencing routines, triggers and applications from performing INSERT and UPDATE operations on those columns. Such changes will be reported at runtime. Similarly, if the view select table expression becomes read-only, referencing queries may fail. SQL considers as read-only views those with select expressions that:
  - ◆ Use the DISTINCT argument to eliminate duplicate rows from the result table
  - ◆ Name more than one table or view in the FROM clause
  - ◆ Use a derived table in the FROM clause
  - ◆ Include a statistical function in the select list
  - ◆ Contain a UNION, EXCEPT DISTINCT (MINUS), INTERSECT DISTINCT, GROUP BY, or HAVING clause
- If the AS clause changes the view to read-only, or includes a LIMIT TO ... ROWS clause on the main query then the check option constraint is implicitly removed.

### *Examples*

A comment can be added or changed on a view using the COMMENT IS clause as shown in this example.

#### *Example 6-1 Changing the comment on a view*

---

```
SQL> show view (comment) current_job
Information for table CURRENT_JOB
```

```

SQL> alter view CURRENT_JOB
cont>   comment is 'Select the most recent job for the employee';
SQL> show view (comment) current_job
Information for table CURRENT_JOB

Comment on table CURRENT_JOB:
Select the most recent job for the employee
SQL>

```

---

The following view uses a derived table and join to collect the count of employees in each department. The view is used in several reporting programs used by the department and company managers.

### *Example 6–2 Changing the columns results of a view definition*

---

```

SQL> create view DEPARTMENTS_SUMMARY
cont> as
cont> select department_code, d.department_name,
cont>        d.manager_id, jh.employee_count
cont> from departments d inner join
cont>        (select department_code, count (*)
cont>         from job_history
cont>         where job_end is null
cont>         group by department_code)
cont>        as jh (department_code, employee_count)
cont>        using (department_code);
SQL>
SQL> show view DEPARTMENTS_SUMMARY;
Information for table DEPARTMENTS_SUMMARY

Columns for view DEPARTMENTS_SUMMARY:
Column Name                Data Type                Domain
-----
DEPARTMENT_CODE            CHAR(4)
DEPARTMENT_NAME            CHAR(30)
Missing Value: None
MANAGER_ID                 CHAR(5)
Missing Value:
EMPLOYEE_COUNT             INTEGER
Source:
select department_code, d.department_name,
        d.manager_id, jh.employee_count
from departments d inner join
        (select department_code, count (*)
         from job_history
         where job_end is null
         group by department_code) as jh (department_code, employee_count)
        using (department_code)
SQL>

```

The database administrator decides to create a column in the DEPARTMENTS table to hold the count of employees (rather than using a query to gather the total) and to maintain the value through triggers on EMPLOYEES and JOB\_HISTORY (not shown here). Now the view can be simplified without resorting to a DROP VIEW and CREATE VIEW. Alter view will preserve the dependencies on the view from other views, triggers and routines and so minimize the work required to implement such a change.

```

SQL> alter table DEPARTMENTS
cont>   add column EMPLOYEE_COUNT integer;

```

```

SQL>
SQL> alter view DEPARTMENTS_SUMMARY
cont> as
cont> select department_code, d.department_name,
cont>         d.manager_id, d.employee_count
cont> from departments d;
SQL>
SQL> show view DEPARTMENTS_SUMMARY;
Information for table DEPARTMENTS_SUMMARY

Columns for view DEPARTMENTS_SUMMARY:
Column Name          Data Type          Domain
-----
DEPARTMENT_CODE      CHAR(4)
Missing Value: None
DEPARTMENT_NAME      CHAR(30)
Missing Value: None
MANAGER_ID            CHAR(5)
Missing Value:
EMPLOYEE_COUNT        INTEGER
Source:
select department_code, d.department_name,
       d.manager_id, d.employee_count
from departments d

```

---

This example shows that a **WITH CHECK OPTION** constraint restricts the inserted data to the view's **WHERE** clause. Once the constraint is removed, the **INSERT** is no longer constrained.

### ***Example 6–3 Changing the WITH CHECK OPTION constraint of a view definition***

---

```

SQL> create view TOLIVER_EMPLOYEE
cont> as select * from EMPLOYEES where employee_id = '00164'
cont> with check option;
SQL> insert into TOLIVER_EMPLOYEE (employee_id) value ('00000');
%RDB-E-INTEG_FAIL, violation of constraint TOLIVER_EMPLOYEE_CHECKOPT1 caused
operation to fail
-RDB-F-ON_DB, on database DISK1:[DATABASES]MF_PERSONNEL.RDB;1
SQL>
SQL> alter view TOLIVER_EMPLOYEE with no check option;
SQL>
SQL> insert into TOLIVER_EMPLOYEE (employee_id) value ('00000');
1 row inserted
SQL>

```

---

# **Chapter 7**

## **Documentation Corrections, Additions and Changes**

This chapter provides corrections for documentation errors and omissions.

# 7.1 Documentation Corrections

## 7.1.1 Incorrect Example Under RMU Unload Command

Bug 4430799

The Oracle RMU Reference Manual, Version 7.1, contains a misleading example. Example 14 in the RMU Unload Command describes a mechanism for reloading AUTOMATIC column data into the database. It should be replaced by this revised example:

AUTOMATIC columns are evaluated during INSERT and UPDATE operations for a table; for instance, they may record the timestamp for the last operation. If the table is being reorganized, it may be necessary to unload the data and reload it after the storage map and indexes for the table are re-created, yet the old auditing data must remain the same.

Normally, RMU Unload does not unload columns marked as AUTOMATIC. You must use the /VIRTUAL\_FIELD qualifier with the keyword AUTOMATIC to request this action.

```
$ rmu/unload/virtual_fields=(automatic) payroll_db people people.unl
```

Following the restructure of the database, the data can be reloaded. If the target columns are also defined as AUTOMATIC, then RMU Load will not write to those columns, therefore, you must use the /VIRTUAL\_FIELD qualifier with the keyword AUTOMATIC to request this action.

```
$ rmu/load/virtual_fields=(automatic) payroll_db people people.unl
```

## 7.1.2 RDM\$BIND\_MAX\_DBR\_COUNT Documentation Clarification

Bugs 1495227 and 3916606

The Rdb7 Guide to Database Performance and Tuning Manual, Volume 2, page A-18, incorrectly describes the use of the RDM\$BIND\_MAX\_DBR\_COUNT logical.

Following is an updated description. Note that the difference in actual behavior between what is in the existing documentation and software is that the logical name only controls the number of database recovery processes created at once during "node failure" recovery (that is, after a system or monitor crash or other abnormal shutdown) for each database.

When an entire database is abnormally shut down (due, for example, to a system failure), the database will have to be recovered in a "node failure" recovery mode. This recovery will be performed by another monitor in the cluster if the database is opened on another node or will be performed the next time the database is opened.

The RDM\$BIND\_MAX\_DBR\_COUNT logical name and the RDB\_BIND\_MAX\_DBR\_COUNT configuration parameter define the maximum number of database recovery (DBR) processes to be simultaneously invoked by the database monitor for each database during a "node failure" recovery.

This logical name and configuration parameter apply only to databases that do not have global buffers enabled. Databases that utilize global buffers have only one recovery process started at a time during a "node failure" recovery.

In a node failure recovery situation with the Row Cache feature enabled (regardless of the global buffer state), the database monitor will start a single database recovery (DBR) process to recover the Row Cache Server (RCS) process and all user processes from the oldest active checkpoint in the database.

---

#### Per-Database Value

*The **RDM\$BIND\_MAX\_DBR\_COUNT** logical name specifies the maximum number of database recovery processes to run at once for each database. For example, if there are 10 databases being recovered and the value for the **RDM\$BIND\_MAX\_DBR\_COUNT** logical name is 8, up to 80 database recovery processes would be started by the monitor after a node failure.*

---

The **RDM\$BIND\_MAX\_DBR\_COUNT** logical name is translated when the monitor process opens a database. Databases should be closed and reopened for a new value of the logical to become effective.

### 7.1.3 Database Server Process Priority Clarification

By default, the database servers (ABS, ALS, DBR, LCS, LRS, RCS) created by the Rdb monitor inherit their VMS process scheduling base priority from the Rdb monitor process. The default priority for the Rdb monitor process is 15.

Individual server priorities can be explicitly controlled via system-wide logical names as described in [Table 7-1](#).

*Table 7-1 Server Process Priority Logical Names*

Logical Name	Use
<b>RDM\$BIND_ABS_PRIORITY</b>	Base Priority for the ABS Server process
<b>RDM\$BIND_ALS_PRIORITY</b>	Base Priority for the ALS Server process
<b>RDM\$BIND_DBR_PRIORITY</b>	Base Priority for the DBR Server process
<b>RDM\$BIND_LCS_PRIORITY</b>	Base Priority for the LCS Server process
<b>RDM\$BIND_LRS_PRIORITY</b>	Base Priority for the LRS Server process
<b>RDM\$BIND_RCS_PRIORITY</b>	Base Priority for the RCS Server process

When the Hot Standby feature is installed, the **RDMAIJSERVER** account is created specifying an account priority of 15. The priority of AIJ server processes on your system can be restricted with the system-wide logical name **RDM\$BIND\_AIJSRV\_PRIORITY**. If this logical name is defined to a value less than 15, an AIJ server process will adjust its base priority to the value specified when the AIJ server process starts. Values from 0 to 31 are allowed for **RDM\$BIND\_AIJSRV\_PRIORITY**, but the process is not able to raise its priority above the **RDMAIJSERVER** account value.

For most applications and systems, Oracle discourages changing the server process priorities.

## 7.1.4 Explanation of SQL\$INT in a SQL Multiversion Environment and How to Redefine SQL\$INT

Bug 2500594

In an environment running multiple versions of Oracle Rdb, for instance Rdb V7.0 and Rdb V7.1, there are now several variant SQL images, such as SQL\$70.EXE and SQL\$71.EXE. However, SQL\$INT.EXE is not variant but acts as a dispatcher using the translation of the logical name RDMS\$VERSION\_VARIANT to activate the correct SQL runtime environment. This image is replaced when a higher version of Oracle Rdb is installed. Thus, using the example above, when Rdb V7.1 is installed, SQL\$INT.EXE will be replaced with the V7.1 SQL\$INT.EXE.

If an application is linked in this environment (using V7.1 SQL\$INT) and the corresponding executable deployed to a system running Oracle Rdb V7.0 multiversion only, the execution of the application may result in the following error:

```
%IMGACT-F-SYMVECMIS, shareable image's symbol vector table mismatch
```

In order to avoid such a problem, the following alternative is suggested:

In the multiversion environment running both Oracle Rdb V7.0 and Oracle Rdb V7.1, run Oracle Rdb V7.0 multiversion by running the command procedures RDB\$SETVER.COM 70 and RDB\$SETVER RESET. This will set up the necessary logical names and symbols that establish the Oracle Rdb V7.0 environment.

For example:

```
$ @SYS$LIBRARY:RDB$SETVER 70

Current PROCESS Oracle Rdb environment is version V7.0-63 (MULTIVERSION)
Current PROCESS SQL environment is version V7.0-63 (MULTIVERSION)
Current PROCESS Rdb/Dispatch environment is version V7.0-63 (MULTIVERSION)

$ @SYS$LIBRARY:RDB$SERVER RESET
```

Now run SQL and verify that the version is correct:

```
$ sql$
SQL> show version
Current version of SQL is: Oracle Rdb SQL V7.0-63
```

Define SQL\$INT to point to the variant SQL\$SHR.EXE. Then, create an options file directing the linker to link with this newly defined SQL\$INT. An example follows:

```
$ DEFINE SQL$INT SYS$SHARE:SQL$SHR'RDMS$VERSION_VARIANT'.EXE
$ LINK TEST_APPL,SQL$USER/LIB,SYS$INPUT/option
SQL$INT/SHARE
^Z
```

The executable is now ready to be deployed to the Oracle Rdb V7.0 multiversion environment and should run successfully.

Please note that with each release of Oracle Rdb, new entry points are added to the SQL\$INT shareable image. This allows the implementation of new functionality. Therefore, applications linked with SQL\$INT from Oracle Rdb V7.1 cannot be run on systems with only Oracle Rdb V7.0 installed. This is because the shareable image does not contain sufficient entry points.

The workaround presented here allows an application to explicitly link with the Oracle Rdb V7.0 version of the image. Such applications are upward compatible and will run on Oracle Rdb V7.0 and Oracle Rdb V7.1. The applications should be compiled and linked under the lowest version.

In environments where Oracle Rdb V7.1 is installed, this workaround is not required because the SQL\$INT image will dynamically activate the appropriate SQL\$SHRxx image as expected.

## 7.1.5 Documentation Omitted Several Reserved Words

Bug 2319321

The following keywords are considered reserved words in Oracle Rdb Release 7.1.

- UID
- CURRENT\_UID
- SYSTEM\_UID
- SESSION\_UID
- RAW
- LONG
- DBKEY
- ROWID
- SYSDATE

In particular, any column which has these names will be occluded by the keyword. i.e. selecting from column UID will be interpreted as referencing the built in function UID and so return a different result.

The correction to this problem is to enable keyword quoting using SET QUOTING RULES 'SQL92' (or 'SQL99') and enclose the column name in quotations.

In addition, SQL will now generate a warning if these reserved words are used (unquoted) in CREATE and ALTER operations.

## 7.1.6 Using Databases from Releases Earlier Than V6.0

Bug 2383967

You cannot convert or restore databases earlier than V6.0 directly to V7.1. The RMU Convert command for V7.1 supports conversions from V6.0 through V7.0 only. If you have a V3.0 through V5.1 database, you must convert it to at least V6.0 and then convert it to V7.1. For example, if you have a V4.2 database, convert it first to at least V6.0, then convert the resulting database to V7.1.

If you attempt to convert a database created prior to V6.0 directly to V7.1, Oracle RMU generates an error.

## 7.1.7 New RMU/BACKUP Storage Area Assignment With Thread Pools

This is to clarify how storage areas are assigned to disk and tape devices using the new RMU/BACKUP THREAD POOL and BACKUP TO MULTIPLE DISK DEVICES features introduced in Oracle Rdb Release 7.1.

For the case of backup to multiple disk devices using thread pools, the algorithm used by RMU/BACKUP to assign threads is to calculate the size of each area as the product of the page length in bytes times the highest page number used (maximum page number) for that area. The area sizes are then sorted by descending size and ascending device name. For internal processing reasons, the system area is placed as the first area in the first thread. Each of the remaining areas is added to whichever thread has the lowest byte count. In this way, the calculated area sizes are balanced between the threads.

For tape devices, the same algorithm is used but the areas are partitioned among writer threads, not disk devices.

The partitioning for backup to multiple disk devices is done by disk device, not by output thread, because there will typically be more disk devices than output threads, and an area can not span a device.

## 7.1.8 RDM\$BIND\_LOCK\_TIMEOUT\_INTERVAL Overrides the Database Parameter

Bug 2203700

When starting a transaction, there are three different values that are used to determine the lock timeout interval for that transaction. Those values are:

1. The value specified in the SET TRANSACTION statement
2. The value stored in the database as specified in CREATE or ALTER DATABASE
3. The value of the logical name RDM\$BIND\_LOCK\_TIMEOUT\_INTERVAL

The timeout interval for a transaction is the smaller of the value specified in the SET TRANSACTION statement and the value specified in CREATE DATABASE. However, if the logical name RDM\$BIND\_LOCK\_TIMEOUT\_INTERVAL is defined, the value of this logical name overrides the value specified in CREATE DATABASE.

The description of how these three values interact, found in several different parts of the Rdb documentation set, is incorrect and will be replaced by the description above.

The lock timeout value in the database can be dynamically modified from the Locking Dashboard in RMU/SHOW STATISTICS. The Per-Process Locking Dashboard can be used to dynamically override the logical name RDM\$BIND\_LOCK\_TIMEOUT\_INTERVAL for one or more processes.

## 7.1.9 New Request Options for RDO, RDBPRE and RDB\$INTERPRET

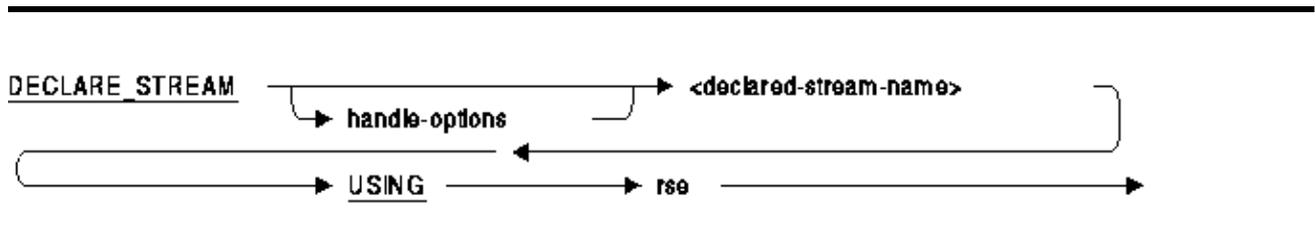
This release note was included in the V70A Release Notes but had gotten dropped somewhere along the line.

For this release of Rdb, two new keywords have been added to the handle-options for the DECLARE\_STREAM, the START\_STREAM (undeclared format) and FOR loop statements. These changes have been made to RDBPRE, RDO and RDB\$INTERPRET at the request of several RDO customers.

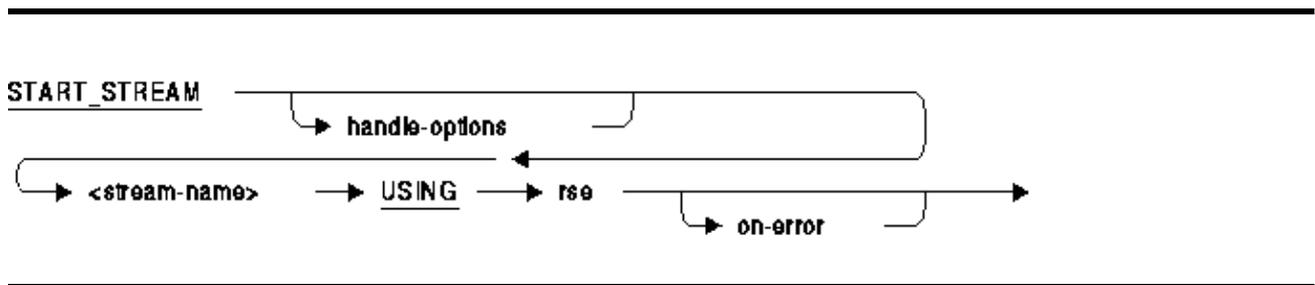
In prior releases, the handle-options could not be specified in interactive RDO or RDB\$INTERPRET. This has changed in Rdb7 but these allowed options will be limited to MODIFY and PROTECTED keywords. For RDBPRE, all options listed will be supported. These option names were chosen to be existing keywords to avoid adding any new keywords to the RDO language.

The altered statements are shown in Example 5-1, Example 5-2 and Example 5-3.

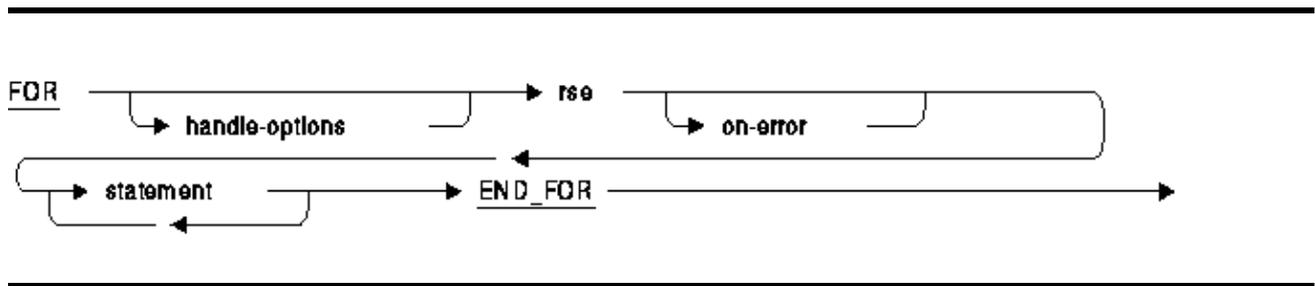
Example 5-1 DECLARE\_STREAM Format



Example 5-2 START\_STREAM Format

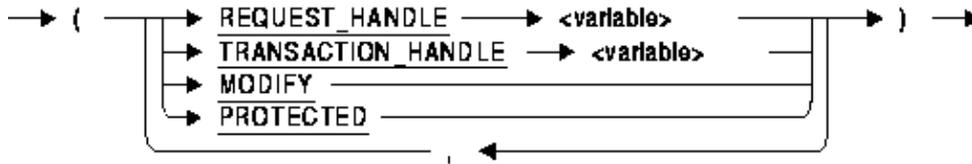


Example 5-3 FOR Format



Each of these statements references the syntax for the HANDLE-OPTIONS which has been revised and is shown below.

handle-options =



The following options are available for HANDLE-OPTIONS:

- REQUEST\_HANDLE specifies the request handle for this request. This option is only valid for RDBPRE and RDML applications. It cannot be used with RDB\$INTERPRET, nor interactive RDO.
- TRANSACTION\_HANDLE specifies the transaction handle under which this request executes. This option is only valid for RDBPRE and RDML applications. It cannot be used with RDB\$INTERPRET, nor interactive RDO.
- MODIFY specifies that the application will modify all (or most) records fetched from the stream or for loop. This option can be used to improve application performance by avoiding lock promotion from SHARED READ for the FETCH to PROTECTED WRITE access for the nested MODIFY or ERASE statement. It can also reduce DEADLOCK occurrence because lock promotions are avoided. This option is valid for RDBPRE, RDB\$INTERPRET, and interactive RDO. This option is not currently available for RDML.

For example:

```
RDO> FOR (MODIFY) E IN EMPLOYEES WITH E.EMPLOYEE_ID = "00164"
cont>     MODIFY E USING E.MIDDLE_INITIAL = "M"
cont>     END_MODIFY
cont> END_FOR
```

This FOR loop uses the MODIFY option to indicate that the nested MODIFY is an unconditional statement and so aggressive locking can be undertaken during the fetch of the record in the FOR loop.

- PROTECTED specifies that the application may modify records fetched by this stream by a separate and independent MODIFY statement. Therefore, this stream should be protected from interference (aka Halloween affect). The optimizer will select a snapshot of the rows and store them in a temporary relation for processing, rather than traversing indexes at the time of the FETCH statement. In some cases this may result in poorer performance when the temporary relation is large and overflows from virtual memory to a temporary disk file, but the record stream will be protected from interference. The programmer is directed to the documentation for the Oracle Rdb logical names RDMS\$BIND\_WORK\_VM and RDMS\$BIND\_WORK\_FILE.

This option is valid for RDBPRE, RDB\$INTERPRET, and interactive RDO. This option is not currently available for RDML.

The following example creates a record stream in a BASIC program using Callable RDO:

```
RDMS_STATUS = RDB$INTERPRET ('INVOKE DATABASE PATHNAME "PERSONNEL"')

RDMS_STATUS = RDB$INTERPRET ('START_STREAM (PROTECTED) EMP USING ' + &
                              'E IN EMPLOYEES')

RDMS_STATUS = RDB$INTERPRET ('FETCH EMP')

DML_STRING = 'GET ' + &
```

## Oracle® Rdb for OpenVMS

```
'!VAL = E.EMPLOYEE_ID;' + &  
'!VAL = E.LAST_NAME;' + &  
'!VAL = E.FIRST_NAME' + &  
'END_GET'
```

```
RDMS_STATUS = RDB$INTERPRET (DML_STRING, EMP_ID, LAST_NAME, FIRST_NAME)
```

In this case the FETCH needs to be protected against MODIFY statements which execute in other parts of the application.

## 7.2 Address and Phone Number Correction for Documentation

In release 7.0 or earlier documentation, the address and fax phone number listed on the Send Us Your Comments page are incorrect. The correct information is:

FAX -- 603.897.3825  
Oracle Corporation  
One Oracle Drive  
Nashua, NH 03062-2804  
USA

## 7.3 Online Document Format and Ordering Information

You can view the documentation in Adobe Acrobat format using the Acrobat Reader, which allows anyone to view, navigate, and print documents in the Adobe Portable Document Format (PDF). See <http://www.adobe.com> for information about obtaining a free copy of Acrobat Reader and for information on supported platforms.

The Oracle Rdb documentation in Adobe Acrobat format is available on MetaLink:

Top Tech Docs\Oracle Rdb\Documentation\`<bookname>`

Customers should contact their Oracle representative to purchase printed documentation.

## 7.4 New and Changed Features in Oracle Rdb Release 7.1

This section provides information about late-breaking new features or information that is missing or changed since the Oracle Rdb New and Changed Features for Oracle Rdb manual was published.

### 7.4.1 PERSONA is Supported in Oracle SQL/Services

In the "New and Changed Features for Oracle Rdb" Manual under the section "ALTER DATABASE Statement" is a note stating that impersonation is not supported in Oracle SQL/Services. This is incorrect. There was a problem in the first release of Oracle Rdb 7.1 (7.1.0) whereby impersonation through Oracle SQL/Services failed. This problem is resolved in Oracle Rdb Release 7.1.0.1.

### 7.4.2 NEXTVAL and CURRVAL Pseudocolumns Can Be Delimited Identifiers

The New and Changed Features for Oracle Rdb manual describes SEQUENCES but does not mention that the special pseudocolumns NEXTVAL and CURRVAL can be delimited. All uppercase and lowercase variations of these keywords are accepted and assumed to be equivalent to these uppercase keywords.

The following example shows that any case is accepted:

```
SQL> set dialect 'sql92';
SQL> create sequence dept_id;
SQL> select dept_id.nextval from rdb$database;
      1
1 row selected
SQL> select "DEPT_ID".currval from rdb$database;
      1
1 row selected
SQL> select "DEPT_ID"."CURRVAL" from rdb$database;
      1
1 row selected
SQL> select "DEPT_ID"."nextval" from rdb$database;
      2
1 row selected
SQL> select "DEPT_ID"."CuRrVaL" from rdb$database;
      2
1 row selected
```

### 7.4.3 Only=select\_list Qualifier for the RMU Dump After\_Journal Command

The Oracle Rdb New and Changed Features for Oracle Rdb manual documents the First=select\_list and Last=select\_list qualifiers for the RMU Dump After\_Journal command. Inadvertently missed was the Only=select\_list qualifier.

The First, Last, and Only qualifiers have been added because the Start and End qualifiers are difficult to use

since users seldom know, nor can they determine, the AIJ record number in advance of using the RMU Dump After\_Journal command.

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 deprecated). 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 preceding example, the dump starts when either block 100 or TSN 52 is encountered.

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, use the Only qualifier and 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 TSN 170 will match the Last=TSN=160 criteria.

## 7.5 Oracle Rdb7 and Oracle CODASYL DBMS Guide to Hot Standby Databases

This section provides information that is missing from or changed in V7.0 of the Oracle Rdb7 and Oracle CODASYL DBMS Guide to Hot Standby Databases.

### 7.5.1 Restrictions Lifted on After-Image Journal Files

The Hot Standby software has been enhanced regarding how it handles after-image journal files. Section 4.2.4 in the Oracle Rdb and Oracle CODASYL DBMS Guide to Hot Standby Databases states the following information:

```
If an after-image journal switchover operation is suspended when
replication operations are occurring, you must back up one or more of
the modified after-image journals to add a new journal file.
```

This restriction has been removed. Now, you can add journal files or use the emergency AIJ feature of Oracle Rdb release 7.0 to automatically add a new journal file. Note the following distinctions between adding an AIJ file and adding an emergency AIJ file:

- You can add an AIJ file to the master database and it will be replicated on the standby database. If replication operations are active, the AIJ file is created on the standby database immediately. If replication operations are not active, the AIJ file is created on the standby database when replication operations are restarted.
- You can add emergency AIJ files anytime. If replication operations are active, the emergency AIJ file is created on the standby database immediately. However, because emergency AIJ files are not journaled, starting replication after you create an emergency AIJ will fail. You cannot start replication operations because the Hot Standby software detects a mismatch in the number of after-image journal files on the master compared to the standby database.

If an emergency AIJ file is created on the master database when replication operations are not active, you must perform a master database backup and then restore the backup on the standby database. Otherwise, an AIJSIGNATURE error results.

### 7.5.2 Changes to RMU Replicate After\_Journal ... Buffer Command

The behavior of the RMU Replicate After\_Journal ... 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 will use a minimum of 4096 buffers. The value provided to the Buffers qualifier will be accepted but 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, then 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, then 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 Nonline qualifier is specified, then 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 then 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.

## 7.5.3 Unnecessary Command in the Hot Standby Documentation

There is an unnecessary command documented in the Oracle Rdb and Oracle CODASYL DBMS Guide to Hot Standby Databases manual. The documentation (in Section 2.12 "Step 10: Specify the Network Transport Protocol") says that to use TCP/IP as the network protocol, you must issue the following commands:

```
$ CONFIG UCX AIJSERVER OBJECT
$ UCX SET SERVICE RDMAIJSRV
/PORT=n
/USER_NAME=RDMAIJSERVER
/PROCESS_NAME=RDMAIJSERVER
/FILE=SYS$SYSTEM:rdmajserver_ucx.com
/LIMIT=nn
```

The first of these commands (\$ CONFIG UCX AIJSERVER OBJECT) is unnecessary. You can safely disregard the first line when setting up to use TCP/IP with Hot Standby.

The documentation will be corrected in a future release of Oracle Rdb.

## 7.5.4 Change in the Way RDMAIJ Server is Set Up in UCX

Starting with Oracle Rdb Release 7.0.2.1, the RDMAIJ image became a variant 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 with regard to setting up the RDMAIJSERVER object when using UCX as the network transport protocol. The UCX SET SERVICE command is now similar to the following:

```
$ UCX SET SERVICE RDMAIJ -
  /PORT=port_number -
  /USER_NAME=RDMAIJ -
  /PROCESS_NAME=RDMAIJ -
  /FILE=SYS$SYSTEM:RDMAIJSERVER.com -
  /LIMIT=limit
```

For Oracle Rdb multiversion, the UCX SET SERVICE command is similar to the following:

```
$ UCX SET SERVICE RDMAIJ70 -
```

## Oracle® Rdb for OpenVMS

```
/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. 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 variant image does not impact installations using DECNet since the correct DECNet object is created during the Oracle Rdb installation.

### **7.5.5 CREATE INDEX Operation Supported for Hot Standby**

On Page 1–13 of the Oracle Rdb7 and Oracle CODASYL DBMS 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.

## 7.6 Oracle Rdb7 for OpenVMS Installation and Configuration Guide

This section provides information that is missing from or changed in V7.0 of the Oracle Rdb7 for OpenVMS Installation and Configuration Guide.

### 7.6.1 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". This section 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 Oracle Rdb images with the Resident qualifier.

Note that modifying the parameter GH\_RSRVPGCNT is only required if the RMONSTART.COM or SQL\$STARTUP.COM procedures have been manually modified to install Oracle 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), there is no need to modify the GH\_RSRVPGCNT parameter.

Oracle Corporation recommends 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 GH\_RSRVPGCNT to be a value of zero in order to ensure the highest level of system performance.

### 7.6.2 Prerequisite Software

In addition to the software listed in the Oracle Rdb Installation and Configuration Guide and at the url [http://www.oracle.com/rdb/product\\_info/index.html](http://www.oracle.com/rdb/product_info/index.html), note that the MACRO-32 compiler and the OpenVMS linker are required OpenVMS components in order to install Oracle Rdb on your OpenVMS Alpha system.

The MACRO-32 Compiler for OpenVMS Alpha is a standard component of the OpenVMS Operating System. It is used to compile VAX MACRO assembly language source files into native OpenVMS Alpha object code. During the Oracle Rdb installation procedure, and portions of the installation verification procedure (such as the test for RDBPRE), the MACRO-32 compiler is required.

The OpenVMS linker is a standard component of the OpenVMS Operating System. It is used to link one or more input files into a program image and defines the execution characteristics of the image. The linker will be required for application development and is likewise used by the Oracle Rdb installation procedure and the installation verification procedure.

### 7.6.3 Defining the RDBSERVER Logical Name

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*

## Oracle® Rdb for OpenVMS

These definitions are inconsistent with other command procedures that attempt to reference the RDBSERVERxx.EXE image. Below 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 correct definition of the logical name is as follows: *DEFINE RDBSERVER SYS\$COMMON:<SYSEXE>RDBSERVER70.EXE*

and *DEFINE RDBSERVER SYS\$COMMON:<SYSEXE>RDBSERVER61.EXE*

## 7.7 Guide to Database Design and Definition

This section provides information that is missing from or changed in release 7.0 of the Oracle Rdb7 Guide to Database Design and Definition.

### 7.7.1 Lock Timeout Interval Logical Incorrect

On Page 7–31 of Section 7.4.8 in the Oracle Rdb7 Guide to Database Design and Definition, the RDM\$BIND\_LOCK\_TIMEOUT logical name is referenced incorrectly. The correct logical name is RDM\$BIND\_LOCK\_TIMEOUT\_INTERVAL.

The Oracle Rdb7 Guide to Database Design and Definition will be corrected in a future release.

### 7.7.2 Example 4–13 and Example 4–14 Are Incorrect

Example 4–13 showing vertical partitioning, and Example 4–14, showing vertical and horizontal partitioning, are incorrect. They should appear as follows:

Example 4–13:

```
SQL> CREATE STORAGE MAP EMPLOYEES_1_MAP
cont>     FOR EMPLOYEES
cont>     ENABLE COMPRESSION
cont>     STORE COLUMNS (EMPLOYEE_ID, LAST_NAME, FIRST_NAME,
cont>                     MIDDLE_INITIAL, STATUS_CODE)
cont>                     DISABLE COMPRESSION
cont>                     IN ACTIVE_AREA
cont>     STORE COLUMNS (ADDRESS_DATA_1, ADDRESS_DATA_2, CITY,
cont>                     STATE, POSTAL_CODE)
cont>                     IN INACTIVE_AREA
cont>     STORE IN OTHER_AREA;
```

Example 4–14:

```
SQL> CREATE STORAGE MAP EMPLOYEES_1_MAP2
cont>     FOR EMP2
cont>     STORE COLUMNS (EMPLOYEE_ID, LAST_NAME, FIRST_NAME,
cont>                     MIDDLE_INITIAL, STATUS_CODE)
cont>                     USING (EMPLOYEE_ID)
cont>                     IN ACTIVE_AREA_A WITH LIMIT OF ('00399')
cont>                     IN ACTIVE_AREA_B WITH LIMIT OF ('00699')
cont>                     OTHERWISE IN ACTIVE_AREA_C
cont>     STORE COLUMNS (ADDRESS_DATA_1, ADDRESS_DATA_2, CITY,
cont>                     STATE, POSTAL_CODE)
cont>                     USING (EMPLOYEE_ID)
cont>                     IN INACTIVE_AREA_A WITH LIMIT OF ('00399')
cont>                     IN INACTIVE_AREA_B WITH LIMIT OF ('00699')
cont>                     OTHERWISE IN INACTIVE_AREA_C
cont>     STORE IN OTHER_AREA;
```

## 7.8 Oracle RMU Reference Manual, Release 7.0

This section provides information that is missing from or changed in V7.0 of the Oracle RMU Reference Manual.

### 7.8.1 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 higher and HP 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;
}
```

## Oracle® Rdb for OpenVMS

```

__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

```

## 7.8.2 New Transaction\_Mode Qualifier for Oracle RMU Commands

A new qualifier, `Transaction_Mode`, has been added to the `RMU Copy`, `Move_Area`, `Restore`, and `Restore Only_Root` commands. You can use this qualifier to set the allowable transaction modes for the database root file created by these commands. If you are not creating a root file as part of one of these commands, for example, you are restoring an area, attempting to use this qualifier returns a `CONFLSWIT` error. This qualifier is similar to the `SET TRANSACTION MODE` clause of the `CREATE DATABASE` command in interactive SQL.

The primary use of this qualifier is when you restore a backup file (of the master database) to create a Hot Standby database. Include the `Transaction_Mode` qualifier on the `RMU Restore` command when you create the standby database (prior to starting replication operations). Because only read-only transactions are allowed on the standby database, you should use the `Transaction_Mode=Read_Only` qualifier setting. This setting prevents modifications to the standby database at all times, even when replication operations are not active.

You can specify the following transaction modes for the `Transaction_Mode` qualifier:

```
All
Current
None
[No]Batch_Update
[No]Read_Only
[No]Exclusive
[No]Exclusive_Read
[No]Exclusive_Write
[No]Protected
[No]Protected_Read
[No]Protected_Write
[No]Shared
[No]Shared_Read
[No]Shared_Write
```

Note that `[No]` indicates that the value can be negated. For example, the `NoExclusive_Write` option indicates that exclusive write is not an allowable access mode for this database. If you specify the `Shared`, `Exclusive`, or `Protected` option, Oracle RMU assumes you are referring to both reading and writing in these modes. For example, the `Transaction_Mode=Shared` option indicates that you want both `Shared_Read` and `Shared_Write` as transaction modes. No mode is enabled unless you add that mode to the list or you use the `ALL` option to enable all modes.

You cannot negate the following three options: `All`, which enables all transaction modes; `None`, which disables all transaction modes; and `Current`, which enables all transaction modes that are set for the source database. If you do not specify the `Transaction_Mode` qualifier, Oracle RMU uses the transaction modes enabled for the source database.

You can list one qualifier that enables or disables a particular mode followed by another that does the opposite. For example, `Transaction_Mode=(NoShared_Write, Shared)` is ambiguous because the first value disables `Shared_Write` access while the second value enables `Shared_Write` access. Oracle RMU resolves the ambiguities by first enabling all modes that are enabled by the items in the `Transaction_Mode` list and then disabling those modes that are disabled by items in the `Transaction_Mode` list. The order of items in the list is irrelevant. In the example discussed, `Shared_Read` is enabled and `Shared_Write` is disabled.

The following example shows how to set a newly restored database to allow read-only transactions only. After Oracle RMU executes the command, the database is ready for you to start Hot Standby replication operations.

```
$ RMU/RESTORE/TRANSACTION_MODE=READ_ONLY MF_PERSONNEL.RBF
```

### 7.8.3 RMU Server After\_Journal Stop Command

If database replication is active and you attempt to stop the database AIJ Log Server, Oracle Rdb returns an error. You must stop database replication before attempting to stop the server.

In addition, a new qualifier, `Output=filename`, has been added to the RMU Server After\_Journal Stop command. This optional qualifier allows you to specify the file where the operational log is to be created. The operational log records the transmission and receipt of network messages.

If you do not include a directory specification with the file name, the log file is created in the database root file directory. It is invalid to include a node name as part of the file name specification.

Note that all Hot Standby bugcheck dumps are written to the corresponding bugcheck dump file; bugcheck dumps are not written to the file you specify with the `Output` qualifier.

### 7.8.4 Incomplete Description of Protection Qualifier for RMU Backup After\_Journal Command

The description of the Protection Qualifier for the RMU Backup After\_Journal command is incomplete in the Oracle RMU Reference Manual for Digital UNIX. The complete description is as follows:

The Protection qualifier specifies the system file protection for the backup file produced by the RMU Backup After\_Journal command. If you do not specify the Protection qualifier, the default access permissions are `-rw-r-----` for backups to disk or tape.

Tapes do not allow delete or execute access and the superuser account always has both read and write access to tapes. In addition, a more restrictive class accumulates the access rights of the less restrictive classes.

If you specify the Protection qualifier explicitly, the differences in access permissions applied for backups to tape or disk as noted in the preceding paragraph are applied. Thus, if you specify `Protection=(S,O,G:W,W:R)`, the access permissions on tape becomes `rw-rw-r-`.

### 7.8.5 RMU Extract Command Options Qualifier

A documentation error exists in the description of the `Options=options-list` qualifier of the RMU Extract command. Currently, the documentation states that this qualifier is not applied to output created by the `Items=Volume` qualifier. This is incorrect. Beginning with 6.1 of Oracle Rdb, the behavior of the `Options=options-list` qualifier is applied to output created by the `Items=Volume` qualifier.

### 7.8.6 RDM\$SNAP\_QUIET\_POINT Logical is Incorrect

On page 2-72 of the Oracle RMU Reference Manual, the reference to the `RDM$SNAP_QUIET_POINT` logical is incorrect. The correct logical name is `RDM$BIND_SNAP_QUIET_POINT`.

### 7.8.7 Using Delta Time with RMU Show Statistics Command

Oracle RMU does not support the use of delta time. However, because the OpenVMS platform does, there is a

## Oracle® Rdb for OpenVMS

workaround. You can specify delta time using the following syntax with the RMU Show Statistics command:

```
$ RMU/SHOW STATISTICS/OUTPUT=file-spec/UNTIL=" ' ' f$cvtime (" +7:00") ' "
```

The +7:00 adds 7 hours to the current time.

You can also use "TOMORROW" and "TODAY+n".

This information will be added to the description of the Until qualifier of the RMU Show Statistics command in a future release of the Oracle RMU Reference Manual.

## 7.9 Oracle Rdb7 Guide to Database Performance and Tuning

The following section provides corrected, clarified, or omitted information for the Oracle Rdb7 Guide to Database Performance and Tuning manual.

### 7.9.1 Dynamic OR Optimization Formats

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)].

### 7.9.2 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. The information in the following table supersedes the entries for the RDM\$BIND\_RUJ\_ALLOC\_BLKCNT and RDM\$BIND\_RUJ\_EXTEND\_BLKCNT logical names.

**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.

### 7.9.3 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 Rdb metadata lock is in use. The term client indicates that Rdb is a client of the 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 drop 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 other users of the table.

The lock name for these special locks consist of an encoded 16 byte name. The first 4 bytes contains the leading four bytes of the user name (for system objects the RDB\$ prefix is skipped) followed by three longwords. The lock is displayed in text format first – here will be seen the prefix for the table, routine, or module name; followed by its hexadecimal representation. The text version masks out non-printable characters with a dot (.).

```
waiting for client '....'...EMPL' 4C504D45000000220000000400000055
```

The leftmost value seen in the hexadecimal output contains the name prefix which is easier read in the text field. Then comes a hex number (00000022) which is the id of the object. The id is described below for tables, views, functions, procedures, modules, and sequences.

- For tables and views, the id represents the unique value found in the RDB\$RELATION\_ID column of the RDB\$RELATIONS system relation for the given table.
- For routines (that is functions and procedures), the id represents the unique value found in the RDB\$ROUTINE\_ID column of the RDB\$ROUTINES system relation for the given routine.
- For modules, the id represents the unique value found in the RDB\$MODULE\_ID column of the RDB\$MODULES system relation for the given module.
- For sequences, the id represents the unique value found in the RDB\$SEQUENCE\_ID column of the RDB\$SEQUENCES system relation for the given sequence.

The next value displayed signifies the object type. The following table describes objects and their hexadecimal type values.

**Table 7–2 Objects and Their Hexadecimal Type Value**

Object	Hexadecimal Value
Tables or views	00000004
Modules	00000015
Routines	00000016
Sequences	00000019

The last value in the hexadecimal output represents the lock type. The hexadecimal value 55 indicates this is a client lock and distinct from page and other data structure locks.

The following example shows a "waiting for client lock" message from a Stall Messages screen while the application was processing the EMPLOYEES table from MF\_PERSONNEL. The terminal should be set to 132 characters wide to view the full client lock string.

```
Process.ID Since..... T Stall.reason.....Lock.ID.
27800643:1          waiting for logical area 79 (CW)          16004833
27800507:1 31-OCT-2002 16:05:15.71 W waiting for client '...."...EMPL' 4C504D45000000220000000040
```

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$ROUTINE_ID = 7;
SQL> select RDB$SEQUENCE_NAME from RDB$SEQUENCES where RDB$SEQUENCE_ID = 2;
```

For more detailed lock information, perform the following steps:

- Press the L option from the horizontal menu to display a menu of lock IDs.
- Select the desired lock ID.

## 7.9.4 RDMS\$TTB\_HASH\_SIZE Logical Name

The logical name RDMS\$TTB\_HASH\_SIZE sets the size of the hash table used for temporary tables. If the logical name is not defined, Oracle Rdb uses a default value of 1249.

If you expect that temporary tables will be large (that is, 10K or more rows), use this logical name to adjust the hash table size to avoid long hash chains. Set the value to approximately 1/4 of the expected maximum number of rows for each temporary table. For example, if a temporary table will be populated with 100,000 rows, define this logical name to be 25000. If there are memory constraints on your system, you should define the logical name to be no higher than this value (1/4 of the expected maximum number of rows).

## 7.9.5 Error in Updating and Retrieving a Row by Dbkey Example 3–22

Example 3–22 in Section 3.8.3 that shows how to update and retrieve a row by dbkey is incorrect. The example should appear as follows:

```
SQL> ATTACH 'FILENAME MF_PERSONNEL.RDB';
SQL> --
SQL> -- Declare host variables
SQL> --
SQL> DECLARE :hv_row INTEGER;           -- Row counter
SQL> DECLARE :hv_employee_id ID_DOM;    -- EMPLOYEE_ID field
SQL> DECLARE :hv_employee_id_ind SMALLINT; -- Null indicator variable
SQL> --
SQL> DECLARE :hv_dbkey CHAR(8);         -- DBKEY storage
SQL> DECLARE :hv_dbkey_ind SMALLINT;    -- Null indicator variable
SQL> --
SQL> DECLARE :hv_last_name LAST_NAME_DOM;
SQL> DECLARE :hv_new_address_data_1 ADDRESS_DATA_1_DOM;
SQL> --
SQL> SET TRANSACTION READ WRITE;
SQL> BEGIN
cont> --
cont> -- Set the search value for SELECT
cont> --
cont> SET :hv_last_name = 'Ames';
cont> --
cont> -- Set the NEW_ADDRESS_DATA_1 value
cont> --
cont> SET :hv_new_address_data_1 = '100 Broadway Ave.';
cont> END;
SQL> COMMIT;
SQL> --
SQL> SET TRANSACTION READ ONLY;
SQL> BEGIN
cont> SELECT E.EMPLOYEE_ID, E.DBKEY
cont> INTO :hv_employee_id INDICATOR :hv_employee_id_ind,
cont>        :hv_dbkey INDICATOR :hv_dbkey_ind
cont> FROM EMPLOYEES E
cont> WHERE E.LAST_NAME = :hv_last_name
cont> LIMIT TO 1 ROW;
cont> --
cont> GET DIAGNOSTICS :hv_row = ROW_COUNT;
cont> END;
SQL> COMMIT;
```

```

SQL> --
SQL> SET TRANSACTION READ WRITE RESERVING EMPLOYEES FOR SHARED WRITE;
SQL> BEGIN
cont> IF (:hv_row = 1) THEN
cont>   BEGIN
cont>     UPDATE EMPLOYEES E
cont>       SET E.ADDRESS_DATA_1 = :hv_new_address_data_1
cont>       WHERE E.DBKEY = :hv_dbkey;
cont>     END;
cont> END IF;
cont> END;
SQL> COMMIT;
SQL> --
SQL> -- Display result of change
SQL> --
SQL> SET TRANSACTION READ ONLY;
SQL> SELECT E.*
cont> FROM EMPLOYEES E
cont> WHERE E.DBKEY = :hv_dbkey;
EMPLOYEE_ID  LAST_NAME      FIRST_NAME  MIDDLE_INITIAL
ADDRESS_DATA_1  ADDRESS_DATA_2  CITY
STATE  POSTAL_CODE  SEX  BIRTHDAY  STATUS_CODE
00416      Ames          Louie      A
100 Broadway Ave.      Alton
NH          03809        M        13-Apr-1941  1

1 row selected
SQL>

```

The new example will appear in a future publication of the Oracle Rdb7 Guide to Database Performance and Tuning manual.

## 7.9.6 Error in Calculation of Sorted Index in Example 3–46

Example 3–46 in Section 3.9.5.1 shows the output when you use the RMU Analyze Indexes command and specify the Option=Debug qualifier and the DEPARTMENTS\_INDEX sorted index.

The description of the example did not include the 8 byte dbkey in the calculation of the sorted index. The complete description is as follows:

The entire index (26 records) is located on pages 2 and 3 in logical area 72 and uses 188 bytes of a possible 430 bytes or the node record is 47 percent full. Note that due to index compression, the node size has decreased in size from 422 bytes to 188 bytes and the percent fullness of the node records has dropped from 98 to 47 percent. Also note that the used/avail value in the summary information at the end of the output does not include the index header and trailer information, which accounts for 32 bytes. This value is shown for each node record in the detailed part of the output. The number of bytes used by the index is calculated as follows: the sort key is 4 bytes plus a null byte for a total of 5 bytes. The prefix is 1 byte and the suffix is 1 byte. The prefix indicates the number of bytes in the preceding key that are the same and the suffix indicates the number of bytes that are different from the preceding key. The dbkey pointer to the row is 8 bytes. There are 26 data rows multiplied by 15 bytes for a total of 390 bytes. The 15 bytes include:

- 7 bytes for the sort key: length + null byte + prefix + suffix
- 8 bytes for the dbkey pointer to the row

Add 32 bytes for index header and trailer information for the index node to the 390 bytes for a total of 422

bytes used. Index compression reduces the number of bytes used to 188 bytes used.

The revised description will appear in a future publication of the Oracle Rdb7 Guide to Database Performance and Tuning manual.

## 7.9.7 Documentation Error in Section C.7

The Oracle Rdb Guide to Database Performance And Tuning, Volume 2 contains an error in Section C.7 titled 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, the statistics should be described as show below:

- 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.

## 7.9.8 Missing Tables Descriptions for the RDBEXPERT Collection Class

Appendix B in the Oracle Rdb7 Guide to Database Performance and Tuning describes the event-based data tables in the formatted database for the Oracle Rdb PERFORMANCE and RDBEXPERT collection classes. This section describes the missing tables for the RDBEXPERT collection class.

Table 7-3 shows the TRANS\_TPB table.

*Table 7-3 Columns for Table EPC\$1\_221\_TRANS\_TPB*

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_POINT	DATE VMS	
CLIENT_PC	INTEGER	
STREAM_ID	INTEGER	
TRANS_ID	VARCHAR(16)	
TRANS_ID_STR_ID	INTEGER	STR_ID_DOMAIN
TPB	VARCHAR(127)	
TPB_STR_ID	INTEGER	STR_ID_DOMAIN

Table 7-4 shows the TRANS\_TPB\_ST table. An index is provided for this table. It is defined with column STR\_ID, duplicates are allowed, and the type is sorted.

*Table 7-4 Columns for Table EPC\$1\_221\_TRANS\_TPB\_ST*

Column Name	Data Type	Domain
STR_ID	INTEGER	STR_ID_DOMAIN
SEGMENT_NUMBER	SMALLINT	SEGMENT_NUMBER_DOMAIN
STR_SEGMENT	VARCHAR(128)	

## 7.9.9 Missing Columns Descriptions for Tables in the Formatted Database

Some of the columns were missing from the tables in Appendix B in the Oracle Rdb7 Guide to Database Performance and Tuning. The complete table definitions are described in this section.

Table 7-5 shows the DATABASE table.

*Table 7-5 Columns for Table EPC\$1\_221\_DATABASE*

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_POINT	DATE VMS	
CLIENT_PC	INTEGER	
STREAM_ID	INTEGER	
DB_NAME	VARCHAR(255)	
DB_NAME_STR_ID	INTEGER	STR_ID_DOMAIN
IMAGE_FILE_NAME	VARCHAR(255)	
IMAGE_FILE_NAME_STR_ID	INTEGER	STR_ID_DOMAIN

Table 7-6 shows the REQUEST\_ACTUAL table.

*Table 7-6 Columns for Table EPC\$1\_221\_REQUEST\_ACTUAL*

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_START	DATE VMS	

TIMESTAMP_END	DATE VMS
DBS_READS_START	INTEGER
DBS_WRITES_START	INTEGER
RUJ_READS_START	INTEGER
RUJ_WRITES_START	INTEGER
AIJ_WRITES_START	INTEGER
ROOT_READS_START	INTEGER
ROOT_WRITES_START	INTEGER
BUFFER_READS_START	INTEGER
GET_VM_BYTES_START	INTEGER
FREE_VM_BYTES_START	INTEGER
LOCK_REQS_START	INTEGER
REQ_NOT_QUEUED_START	INTEGER
REQ_STALLS_START	INTEGER
REQ_DEADLOCKS_START	INTEGER
PROM_DEADLOCKS_START	INTEGER
LOCK_RELS_START	INTEGER
LOCK_STALL_TIME_START	INTEGER
D_FETCH_RET_START	INTEGER
D_FETCH_UPD_START	INTEGER
D_LB_ALLOK_START	INTEGER
D_LB_GBNEEDLOCK_START	INTEGER
D_LB_NEEDLOCK_START	INTEGER
D_LB_OLDVER_START	INTEGER
D_GB_NEEDLOCK_START	INTEGER
D_GB_OLDVER_START	INTEGER
D_NOTFOUND_IO_START	INTEGER
D_NOTFOUND_SYN_START	INTEGER
S_FETCH_RET_START	INTEGER
S_FETCH_UPD_START	INTEGER
S_LB_ALLOK_START	INTEGER
S_LB_GBNEEDLOCK_START	INTEGER
S_LB_NEEDLOCK_START	INTEGER
S_LB_OLDVER_START	INTEGER
S_GB_NEEDLOCK_START	INTEGER
S_GB_OLDVER_START	INTEGER
S_NOTFOUND_IO_START	INTEGER
S_NOTFOUND_SYN_START	INTEGER
D_ASYNC_FETCH_START	INTEGER
S_ASYNC_FETCH_START	INTEGER
D_ASYNC_READIO_START	INTEGER
S_ASYNC_READIO_START	INTEGER

AS_READ_STALL_START	INTEGER	
AS_BATCH_WRITE_START	INTEGER	
AS_WRITE_STALL_START	INTEGER	
BIO_START	INTEGER	
DIO_START	INTEGER	
PAGEFAULTS_START	INTEGER	
PAGEFAULT_IO_START	INTEGER	
CPU_START	INTEGER	
CURRENT_PRIO_START	SMALLINT	
VIRTUAL_SIZE_START	INTEGER	
WS_SIZE_START	INTEGER	
WS_PRIVATE_START	INTEGER	
WS_GLOBAL_START	INTEGER	
CLIENT_PC_END	INTEGER	
STREAM_ID_END	INTEGER	
REQ_ID_END	INTEGER	
COMP_STATUS_END	INTEGER	
REQUEST_OPER_END	INTEGER	
TRANS_ID_END	VARCHAR(16)	
TRANS_ID_END_STR_ID	INTEGER	STR_ID_DOMAIN
DBS_READS_END	INTEGER	
DBS_WRITES_END	INTEGER	
RUJ_READS_END	INTEGER	
RUJ_WRITES_END	INTEGER	
AIJ_WRITES_END	INTEGER	
ROOT_READS_END	INTEGER	
ROOT_WRITES_END	INTEGER	
BUFFER_READS_END	INTEGER	
GET_VM_BYTES_END	INTEGER	
FREE_VM_BYTES_END	INTEGER	
LOCK_REQS_END	INTEGER	
REQ_NOT_QUEUED_END	INTEGER	
REQ_STALLS_END	INTEGER	
REQ_DEADLOCKS_END	INTEGER	
PROM_DEADLOCKS_END	INTEGER	
LOCK_RELS_END	INTEGER	
LOCK_STALL_TIME_END	INTEGER	
D_FETCH_RET_END	INTEGER	
D_FETCH_UPD_END	INTEGER	
D_LB_ALLOK_END	INTEGER	
D_LB_GBNEEDLOCK_END	INTEGER	
D_LB_NEEDLOCK_END	INTEGER	

D_LB_OLDVER_END	INTEGER
D_GB_NEEDLOCK_END	INTEGER
D_GB_OLDVER_END	INTEGER
D_NOTFOUND_IO_END	INTEGER
D_NOTFOUND_SYN_END	INTEGER
S_FETCH_RET_END	INTEGER
S_FETCH_UPD_END	INTEGER
S_LB_ALLOK_END	INTEGER
S_LB_GBNEEDLOCK_END	INTEGER
S_LB_NEEDLOCK_END	INTEGER
S_LB_OLDVER_END	INTEGER
S_GB_NEEDLOCK_END	INTEGER
S_GB_OLDVER_END	INTEGER
S_NOTFOUND_IO_END	INTEGER
S_NOTFOUND_SYN_END	INTEGER
D_ASYNC_FETCH_END	INTEGER
S_ASYNC_FETCH_END	INTEGER
D_ASYNC_READIO_END	INTEGER
S_ASYNC_READIO_END	INTEGER
AS_READ_STALL_END	INTEGER
AS_BATCH_WRITE_END	INTEGER
AS_WRITE_STALL_END	INTEGER
BIO_END	INTEGER
DIO_END	INTEGER
PAGEFAULTS_END	INTEGER
PAGEFAULT_IO_END	INTEGER
CPU_END	INTEGER
CURRENT_PRIO_END	SMALLINT
VIRTUAL_SIZE_END	INTEGER
WS_SIZE_END	INTEGER
WS_PRIVATE_END	INTEGER
WS_GLOBAL_END	INTEGER

Table 7-7 shows the TRANSACTION table.

**Table 7-7 Columns for Table EPC\$1\_221\_TRANSACTION**

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_START	DATE VMS	

TIMESTAMP_END	DATE VMS	
CLIENT_PC_START	INTEGER	
STREAM_ID_START	INTEGER	
LOCK_MODE_START	INTEGER	
TRANS_ID_START	VARCHAR(16)	
TRANS_ID_START_STR_ID	INTEGER	STR_ID_DOMAIN
GLOBAL_TID_START	VARCHAR(16)	
GLOBAL_TID_START_STR_ID	INTEGER	STR_ID_DOMAIN
DBS_READS_START	INTEGER	
DBS_WRITES_START	INTEGER	
RUJ_READS_START	INTEGER	
RUJ_WRITES_START	INTEGER	
AIJ_WRITES_START	INTEGER	
ROOT_READS_START	INTEGER	
ROOT_WRITES_START	INTEGER	
BUFFER_READS_START	INTEGER	
GET_VM_BYTES_START	INTEGER	
FREE_VM_BYTES_START	INTEGER	
LOCK_REQS_START	INTEGER	
REQ_NOT_QUEUED_START	INTEGER	
REQ_STALLS_START	INTEGER	
REQ_DEADLOCKS_START	INTEGER	
PROM_DEADLOCKS_START	INTEGER	
LOCK_RELS_START	INTEGER	
LOCK_STALL_TIME_START	INTEGER	
D_FETCH_RET_START	INTEGER	
D_FETCH_UPD_START	INTEGER	
D_LB_ALLOK_START	INTEGER	
D_LB_GBNEEDLOCK_START	INTEGER	
D_LB_NEEDLOCK_START	INTEGER	
D_LB_OLDVER_START	INTEGER	
D_GB_NEEDLOCK_START	INTEGER	
D_GB_OLDVER_START	INTEGER	
D_NOTFOUND_IO_START	INTEGER	
D_NOTFOUND_SYN_START	INTEGER	
S_FETCH_RET_START	INTEGER	
S_FETCH_UPD_START	INTEGER	
S_LB_ALLOK_START	INTEGER	
S_LB_GBNEEDLOCK_START	INTEGER	
S_LB_NEEDLOCK_START	INTEGER	
S_LB_OLDVER_START	INTEGER	
S_GB_NEEDLOCK_START	INTEGER	

S_GB_OLDVER_START	INTEGER	
S_NOTFOUND_IO_START	INTEGER	
S_NOTFOUND_SYN_START	INTEGER	
D_ASYNC_FETCH_START	INTEGER	
S_ASYNC_FETCH_START	INTEGER	
D_ASYNC_READIO_START	INTEGER	
S_ASYNC_READIO_START	INTEGER	
AS_READ_STALL_START	INTEGER	
AS_BATCH_WRITE_START	INTEGER	
AS_WRITE_STALL_START	INTEGER	
AREA_ITEMS_START	VARCHAR(128)	
AREA_ITEMS_START_STR_ID	INTEGER	STR_ID_DOMAIN
BIO_START	INTEGER	
DIO_START	INTEGER	
PAGEFAULTS_START	INTEGER	
PAGEFAULT_IO_START	INTEGER	
CPU_START	INTEGER	
CURRENT_PRIO_START	SMALLINT	
VIRTUAL_SIZE_START	INTEGER	
WS_SIZE_START	INTEGER	
WS_PRIVATE_START	INTEGER	
WS_GLOBAL_START	INTEGER	
CROSS_FAC_2_START	INTEGER	
CROSS_FAC_3_START	INTEGER	
CROSS_FAC_7_START	INTEGER	
CROSS_FAC_14_START	INTEGER	
DBS_READS_END	INTEGER	
DBS_WRITES_END	INTEGER	
RUJ_READS_END	INTEGER	
RUJ_WRITES_END	INTEGER	
AIJ_WRITES_END	INTEGER	
ROOT_READS_END	INTEGER	
ROOT_WRITES_END	INTEGER	
BUFFER_READS_END	INTEGER	
GET_VM_BYTES_END	INTEGER	
FREE_VM_BYTES_END	INTEGER	
LOCK_REQS_END	INTEGER	
REQ_NOT_QUEUED_END	INTEGER	
REQ_STALLS_END	INTEGER	
REQ_DEADLOCKS_END	INTEGER	
PROM_DEADLOCKS_END	INTEGER	
LOCK_RELS_END	INTEGER	

LOCK_STALL_TIME_END	INTEGER	
D_FETCH_RET_END	INTEGER	
D_FETCH_UPD_END	INTEGER	
D_LB_ALLOK_END	INTEGER	
D_LB_GBNEEDLOCK_END	INTEGER	
D_LB_NEEDLOCK_END	INTEGER	
D_LB_OLDVER_END	INTEGER	
D_GB_NEEDLOCK_END	INTEGER	
D_GB_OLDVER_END	INTEGER	
D_NOTFOUND_IO_END	INTEGER	
D_NOTFOUND_SYN_END	INTEGER	
S_FETCH_RET_END	INTEGER	
S_FETCH_UPD_END	INTEGER	
S_LB_ALLOK_END	INTEGER	
S_LB_GBNEEDLOCK_END	INTEGER	
S_LB_NEEDLOCK_END	INTEGER	
S_LB_OLDVER_END	INTEGER	
S_GB_NEEDLOCK_END	INTEGER	
S_GB_OLDVER_END	INTEGER	
S_NOTFOUND_IO_END	INTEGER	
S_NOTFOUND_SYN_END	INTEGER	
D_ASYNC_FETCH_END	INTEGER	
S_ASYNC_FETCH_END	INTEGER	
D_ASYNC_READIO_END	INTEGER	
S_ASYNC_READIO_END	INTEGER	
AS_READ_STALL_END	INTEGER	
AS_BATCH_WRITE_END	INTEGER	
AS_WRITE_STALL_END	INTEGER	
AREA_ITEMS_END	VARCHAR(128)	
AREA_ITEMS_END_STR_ID	INTEGER	STR_ID_DOMAIN
BIO_END	INTEGER	
DIO_END	INTEGER	
PAGEFAULTS_END	INTEGER	
PAGEFAULT_IO_END	INTEGER	
CPU_END	INTEGER	
CURRENT_PRIO_END	SMALLINT	
VIRTUAL_SIZE_END	INTEGER	
WS_SIZE_END	INTEGER	
WS_PRIVATE_END	INTEGER	
WS_GLOBAL_END	INTEGER	
CROSS_FAC_2_END	INTEGER	
CROSS_FAC_3_END	INTEGER	

CROSS_FAC_7_END	INTEGER
CROSS_FAC_14_END	INTEGER

Table 7–8 shows the REQUEST\_BLR table.

*Table 7–8 Columns for Table EPC\$1\_221\_REQUEST\_BLR*

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_POINT	DATE VMS	
CLIENT_PC	INTEGER	
STREAM_ID	INTEGER	
REQ_ID	INTEGER	
TRANS_ID	VARCHAR(16)	
TRANS_ID_STR_ID	INTEGER	STR_ID_DOMAIN
REQUEST_NAME	VARCHAR(31)	
REQUEST_NAME_STR_ID	INTEGER	STR_ID_DOMAIN
REQUEST_TYPE	INTEGER	
BLR	VARCHAR(127)	
BLR_STR_ID	INTEGER	STR_ID_DOMAIN

## 7.9.10 A Way to Find the Transaction Type of a Particular Transaction Within the Trace Database

The table EPC\$1\_221\_TRANSACTION in the formatted Oracle Trace database has a column LOCK\_MODE\_START of longword datatype. The values of this column indicate the type of transaction a particular transaction was.

Value	Transaction type
-----	-----
8	Read only
9	Read write
14	Batch update

## 7.9.11 Using Oracle TRACE Collected Data

The following example shows how the OPTIMIZE AS clause is reflected in the Oracle TRACE database. When a trace collection is started the following SQL commands will record the request names.

```
SQL> attach `file personnel`;
SQL> select last_name, first_name
cont> from employees
```

## Oracle® Rdb for OpenVMS

```
cont> optimize as request_one;
.
.
.
SQL> select employee_id
cont> from employees
cont> optimize as request_two;
.
.
.
SQL> select employee_id, city, state
cont> from employees
cont> optimize as request_three;
.
.
.
SQL> select last_name, first_name, employee_id, city, state
cont> from employees
cont> optimize as request_four;
.
.
.
```

Once an Oracle TRACE database has been populated from the collection, a query such as the following can be used to display the request names and types. The type values are described in Table 3–10. The unnamed queries in this example correspond to the queries executed by interactive SQL to validate the names of the tables and columns referenced in the user supplied queries.

```
SQL> select REQUEST_NAME, REQUEST_TYPE, TIMESTAMP_POINT
cont> from EPC$1_221_REQUEST_BLR;
REQUEST_NAME                REQUEST_TYPE    TIMESTAMP_POINT
                        1  15-JAN-1997 13:23:27.18
                        1  15-JAN-1997 13:23:27.77
REQUEST_ONE                 1  15-JAN-1997 13:23:28.21
REQUEST_TWO                 1  15-JAN-1997 13:23:56.55
REQUEST_THREE               1  15-JAN-1997 13:24:57.27
REQUEST_FOUR                1  15-JAN-1997 13:25:25.44
6 rows selected
```

The next example shows the internal query format (BLR) converted to SQL strings after EPC\$EXAMPLES:EPC\_BLR\_TOSQL\_CONVERTER.COM has been run.

```
SQL> SELECT A.REQUEST_NAME, B.SQL_STRING FROM
cont> EPC$1_221_REQUEST_BLR A,
cont> EPC$SQL_QUERIES B
cont> WHERE A.CLIENT_PC = 0 AND A.SQL_ID = B.SQL_ID;
A.REQUEST_NAME
  B.SQL_STRING
REQUEST_ONE
      SELECT C1.LAST_NAME, C1.FIRST_NAME.          FROM EMPLOYEES C1
. . .
REQUEST_TWO
      SELECT C1.EMPLOYEE_ID.                      FROM EMPLOYEES C1
. . .
REQUEST_THREE
      SELECT C1.EMPLOYEE_ID, C1.CITY, C1.STATE.    FROM EMPLOYEES C1
.
.
.
4 rows selected
```

Table 4–17 shows the Request Types.

**Table 7–9 Request Types**

Symbolic Name	Value	Comment
RDB_K_REQTYPE_OTHER	0	A query executed internally by Oracle Rdb
RDB_K_REQTYPE_USER_REQUEST	1	A non–stored SQL statement, which includes compound statements
RDB_K_REQTYPE_PROCEDURE	2	A stored procedure
RDB_K_REQTYPE_FUNCTION	3	A stored function
RDB_K_REQTYPE_TRIGGER	4	A trigger action
RDB_K_REQTYPE_CONSTRAINT	5	A table or column constraint

## 7.9.12 AIP Length Problems in Indexes that Allow Duplicates

When an index allows duplicates, the length stored in the AIP will be 215 bytes, regardless of the actual index node size. Because an index with duplicates can have variable node sizes, the 215–byte size is used as a median length to represent the length of rows in the index's logical area.

When the row size in the AIP is less than the actual row length, it is highly likely that SPAM entries will show space is available on pages when they have insufficient space to store another full size row. This is the most common cause of insert performance problems.

For example, consider a case where an index node size of 430 bytes (a common default value) is used; the page size for the storage area where the index is stored is 2 blocks. After deducting page overhead, the available space on a 2–block page is 982 bytes. Assume that the page in this example is initially empty.

1. A full size (430–byte) index node is stored. As 8 bytes of overhead are associated with each row stored on a page, that leaves  $982 - 430 - 8 = 544$  free bytes remaining on the page.
2. A duplicate key entry is made in that index node and thus a duplicate node is created on the same page. An initial duplicate node is 112 bytes long (duplicate nodes can have a variety of sizes depending on when they are created, but for this particular example, 112 bytes is used). Therefore,  $544 - 112 - 8 = 424$  free bytes remain on the page.

At this point, 424 bytes are left on the page. That is greater than the 215 bytes that the AIP shows as the row length for the logical area, so the SPAM page shows that the page has space available. However, an attempt to store a full size index node on the page will fail, because the remaining free space (424 bytes) is not enough to store a 430–byte node.

In this case, another candidate page must be selected via the SPAM page, and the process repeats until a page that truly has sufficient free space available is found. In a logical area that contains many duplicate nodes, a significant percentage of the pages in the logical area may fit the scenario just described. When that is the case, and a new full size index node needs to be stored, many pages may need to be read and checked before one is found that can be used to store the row.

It is possible to avoid the preceding scenario by using logical area thresholds. The goal is to set a threshold such that the SPAM page will show a page is full when space is insufficient to store a full size index node.

Using the previous example, here is how to properly set logical area thresholds to prevent excessive pages checked on an index with a 430-byte node size that is stored on a 2-block page. To calculate the proper threshold value to use, you must first determine how full the page can get before no more full size nodes will fit on the page. In this example, a database page can have up to  $982 - 430 - 8 = 544$  bytes in use before the page is too full. Therefore, if 544 or fewer bytes are in use, then enough space remains to store another full size node. The threshold is then  $544 / 982 = .553971$ , or 55%.

In addition, you can determine how full a page must be before a duplicate node of size 112 will no longer fit. In this example, a database page can have up to  $982 - 112 - 8 = 862$  bytes in use before the page is too full. Therefore, if 862 or fewer bytes are in use, then enough space remains to store another small duplicates node. The threshold is then  $862 / 982 = .8778$ , or 88%.

Here is an example of creating an index with the above characteristics:

```
SQL> CREATE INDEX TEST_INDEX ON EMPLOYEES (LAST_NAME)
cont>     STORE IN RDB$SYSTEM
cont>     (THRESHOLD IS (55, 55, 88));
```

These settings mean that any page at over 55% full will not be fetched when inserting a full index node, however, it may be fetched when inserting the smaller duplicates node. When the page is over 88% full then neither a full node nor a duplicate node can be stored, so the page is set as FULL. The lowest setting is not used and so can be set to any value less than or equal to the lowest used threshold.

Note that the compression algorithm used on regular tables that have compression enabled does not apply to index nodes. Index nodes are not compressed like data rows and will always utilize the number of bytes that is specified in the node size. Do not attempt to take into account a compression factor when calculating thresholds for indexes.

## 7.9.13 RDM\$BIND\_MAX\_DBR\_COUNT Documentation Clarification

Appendix A in Oracle Rdb7 Guide to Database Performance and Tuning incorrectly describes the use of the RDM\$BIND\_MAX\_DBR\_COUNT logical name.

Following is an updated description. Note that the difference in actual behavior between what is in the existing documentation and the software is that the logical name only controls the number of database recovery processes created at once during "node failure" recovery (that is, after a system or monitor crash or other abnormal shutdown).

When an entire database is abnormally shut down (due, for example, to a system failure), the database will have to be recovered in a "node failure" recovery mode. This recovery will be performed by another monitor in the cluster if the database is opened on another node or will be performed the next time the database is opened.

The RDM\$BIND\_MAX\_DBR\_COUNT logical name and the RDB\_BIND\_MAX\_DBR\_COUNT configuration parameter define the maximum number of database recovery (DBR) processes to be simultaneously invoked by the database monitor during a "node failure" recovery.

## Oracle® Rdb for OpenVMS

This logical name and configuration parameter apply only to databases that do not have global buffers enabled. Databases that utilize global buffers have only one recovery process started at a time during a "node failure" recovery.

In a node failure recovery situation with the Row Cache feature enabled (regardless of the global buffer state), the database monitor will start a single database recovery (DBR) process to recover the Row Cache Server (RCS) process and all user processes from the oldest active checkpoint in the database.

## 7.10 Oracle Rdb7 Guide to SQL Programming

This section provides information that is missing or changed in the Oracle Rdb7 Guide to SQL Programming.

### 7.10.1 Location of Host Source File Generated by the SQL Precompiler

When the SQL precompiler generates host source files (for example, .c, .pas, or .for) from the precompiler source files, it locates these files based on the Object qualifier in the command given to the SQL precompiler.

The following examples show the location where the host source file is generated.

When the Object qualifier 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. For example:

```
$ sqlpre/cc scc_try_mli_successful.sc
$ 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 the Object qualifier 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 a language other than C, it uses the appropriate host source extension (for example, .pas or .for). The files also default to the current directory if a directory specification is not specified. For example:

```
$ sqlpre/cc/obj=myobj scc_try_mli_successful.sc
$ dir scc_try_mli_successful.*

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 1 file.
$ dir myobj.*

Directory MYDISK:[LUND]

MYOBJ.C;1          MYOBJ.OBJ;2

Total of 2 files.

$ sqlpre/cc/obj=MYDISK:[lund.tmp] scc_try_mli_successful.sc
$ dir scc_try_mli_successful.*

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 1 file.
```

```

$ 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.

```

## 7.10.2 Remote User Authentication

In the Oracle Rdb7 Guide to SQL Programming, Table 15–1 indicates that implicit authorization works from an OpenVMS platform to another OpenVMS platform using TCP/IP. This table is incorrect. Implicit authorization only works using DECnet in this situation.

The Oracle Rdb7 Guide to SQL Programming will be fixed in a future release.

## 7.10.3 Additional Information About Detached Processes

Oracle Rdb documentation omits necessary detail on running Oracle Rdb from a detached process.

Applications run from detached processes must ensure that the OpenVMS environment is established correctly before running Oracle Rdb, otherwise Oracle Rdb will not execute.

Attempts to attach to a database and execute an Oracle Rdb query from applications running as detached processes will result in an error similar to the following:

```

%RDB-F-SYS_REQUEST, error from system services request
-SORT-E-OPENOUT, error opening [file] as output
-RMS-F-DEV, error in device name or inappropriate device type for operation

```

The problem occurs because a detached process does not normally have the logical names SYSS\$LOGIN or SYSS\$SCRATCH defined.

There are two methods that can be used to correct this:

- Solution 1:

Use the DCL command procedure RUN\_PROCEDURE to run the ACCOUNTS application:

RUN\_PROCEDURE.COM includes the single line:

```
$ RUN ACCOUNTS_REPORT
```

Then execute this procedure using this command:

```
$ RUN/DETACH/AUTHORIZE SYSS$SYSTEM:LOGINOUT/INPUT=RUN_PROCEDURE
```

This solution executes SYSS\$SYSTEM:LOGINOUT so that the command language interface (DCL) is activated. This causes the logical names SYSS\$LOGIN and SYSS\$SCRATCH to be defined for the detached process. The /AUTHORIZE qualifier also ensures that the users' process quota limits (PQLs) are used from the system authorization file rather than relying on the default PQL system parameters, which are often insufficient to run Oracle Rdb.

- Solution 2:

If DCL is not desired, and SYSS\$LOGIN and SYSS\$SCRATCH are not defined, then prior to executing any Oracle Rdb statement, you should define the following logical names:

- ◆ RDMS\$BIND\_WORK\_FILE

## Oracle® Rdb for OpenVMS

Define this logical name to allow you to reduce the overhead of disk I/O operations for matching operations when used in conjunction with the `RDMS$BIND_WORK_VM` logical name. If the virtual memory file is too small then overflow to disk will occur at the disk and directory location specified by `RDMS$BIND_WORK_FILE`.

For more information on `RDMS$BIND_WORK_FILE` and `RDMS$BIND_WORK_VM`, see the Oracle Rdb Guide to Database Performance and Tuning.

- ◆ `SORTWORK0`, `SORTWORK1`, and so on

The OpenVMS Sort/Merge utility (`SORT/MERGE`) attempts to create sort work files in `SYSSCRATCH`. If the `SORTWORK` logical names exist, the utility will not require the `SYSSCRATCH` logical. However, note that not all queries will require sorting, and that some sorts will be completed in memory and so will not necessarily require disk space.

If you use the logical `RDMS$BIND_SORT_WORKFILES`, you will need to define further `SORTWORK` logical names as described in the Oracle Rdb Guide to Database Performance and Tuning.

You should also verify that sufficient process quotas are specified on the `RUN/DETACH` command line, or defined as system PQL parameters to allow Oracle Rdb to execute.

# 7.11 Guide to Using Oracle SQL/Services Client APIs

The following information describes Oracle SQL/Services documentation errors or omissions.

- The Guide to Using Oracle SQL/Services Client APIs does not describe changes to size and format of integer and floating-point data types  
Beginning with Oracle SQL/Services V5.1, the size and format of some integer and floating-point data types is changed as follows:

- ◆ Trailing zeros occur in fixed-point numeric data types with SCALE FACTOR.  
Trailing zeros are now included after the decimal point up to the number of digits specified by the SCALE FACTOR. In versions of Oracle SQL/Services previous to V5.1, at most one trailing zero was included where the value was a whole number.  
The following examples illustrate the changes using a field defined as INTEGER(3):

V5.1 and higher	Versions previous to V5.1
1.000	1.0
23.400	23.4
567.890	567.89

- ◆ Trailing zeros occur in floating-point data types. Trailing zeros are now included in the fraction, and leading zeros are included in the exponent, up to the maximum precision available, for fields assigned the REAL and DOUBLE PRECISION data types.

Data Type	V5.1 and higher	Versions previous to V5.1
REAL	1.2340000E+01	1.234E+1
DOUBLE PRECISION	5.6789000000000000E+001	5.6789E+1

- ◆ Size of TINYINT and REAL data types is changed.  
The maximum size of the TINYINT and REAL data types is changed to correctly reflect the precision of the respective data types.  
The following table shows the maximum lengths of the data types now and in previous versions:

Data type	V5.1 and higher	Versions previous to V5.1
TINYINT	4	6
REAL	15	24

- The Guide to Using Oracle SQL/Services Client APIs does not describe that the sqlsrv\_associate() service returns SQL error code -1028 when connecting to a database service if the user has not been granted the right to attach to the database.  
When a user connects to a database service, the sqlsrv\_associate() service completes with the SQL error code -1028, SQL\_NO\_PRIV, if the user has been granted access to the Oracle SQL/Services service, but has not been granted the right to attach to the database. A record of the failure is written to the executor process's log file. Note that the sqlsrv\_associate() service completes with the Oracle SQL/Services error code -2034, SQLSRV\_GETACCINF if the user has not been granted access to the Oracle SQL/Services service.

# Chapter 8

## Known Problems and Restrictions

This chapter describes problems and restrictions relating to Oracle Rdb Release 7.1.4.3, and includes workarounds where appropriate.

## 8.1 Known Problems and Restrictions in All Interfaces

This section describes known problems and restrictions that affect all interfaces for Release 7.1.

### 8.1.1 Domain-qualified TCP/IP Node Names in Distributed Transactions

Bug 3735144

When using TCP/IP for Oracle Rdb remote connections, distributed transactions involving databases on nodes which are not on the same subnet may not work.

Remote Rdb has the capability to make remote connections via TCP/IP in lieu of DECnet. (See the Oracle Rdb OpenVMS Installation and Configuration Guide for how to set this up.) However, distributed transactions involving remote databases connected via TCP/IP have been difficult. This is because Rdb relies on OpenVMS DECdtm for distributed transaction support and DECdtm requires DECnet for off-node communication. (This is an OpenVMS and not an Rdb restriction. Contact Hewlett-Packard OpenVMS Support for more details.)

OpenVMS provides a capability to run DECnet over TCP/IP so that OpenVMS services which require DECnet (like DECdtm) can operate in an environment where a TCP/IP network is used as the communications backbone. This capability allows DECdtm (and hence Rdb) to manage distributed transactions via TCP/IP. (See HP's OpenVMS DECnet-Plus documentation set for how to configure and use this capability.)

However, for a transaction involving a remote database, Rdb only provides the SCSNODE name of the remote node to DECdtm. For example, consider the following SQL attaches to two remote databases using TCP/IP:

```
SQL>attach 'alias db1 filename node1.a.b.c::db_root:db1 user 'me' using
'pw';
SQL>attach 'alias db2 filename node1.a.b.c::db_root:db2 user 'me' using
'pw';
```

In the above example, Rdb can successfully connect to both remote databases using the TCP/IP address "node1.a.b.c." but when multiple databases are attached, Rdb implicitly uses distributed transactions via DECdtm. Since Rdb only passes DECdtm the SCSNODE name retrieved from the RDBSERVERnn at the other end of the connection, DECdtm does not, in general, have the information it needs to resolve the remote reference. It will only be able to do so if the SCSNODE name and the TCP/IP node name are the same and the local node is on the same subnet (i.e. ".a.b.c" in the example). Otherwise, after the second attach is made, the following error message will be received as soon as a transaction is started:

```
SQL>set trans read write;
%RDB-F-SYS_REQUEST_CAL, error from system services request - called from 100001
-RDB-E-DECDTMERR, DECdtm system service call error
-IPC-E-BCKTRNSFAIL, failure on the back translate address request
```

#### **WORKAROUND**

There are three potential workarounds:

1. If distributed transactions are unimportant to the application, they can be disabled by defining the logical name `SQL$DISABLE_CONTEXT` to `TRUE`. Rdb will then not call `DECdtm` and the node name resolution problem will not be seen. However, it will be the problem of the application to maintain database integrity in the event that a commit succeeds on one database and not on another. See the Rdb Guide to Distributed Transactions for more information.
2. If all the nodes involved in the distributed transaction are in the same domain, then TCP/IP can resolve the node with only the first part of the node provided that the `SCSNODE` name is identical to it. In the example above, this would mean that the remote node had an `SCSNODE` name of "NODE1" and that the local node was on TCP/IP subnet ".a.b.c".
3. It may also be possible to define a DNS/BIND alias name for the remote node's `SCSNODE` name to the local node's TCP/IP database. This should allow the `SCSNODE` name passed by Rdb Dispatch to be translated successfully. For example, assuming HP TCP/IP Services for OpenVMS is the TCP/IP protocol stack then a command like the following could be used on the local node:

```
$ TCP SET HOST NODE1.A.B.C/address=nnn.nnn.nnn.nnn/alias=NODE1_SCS
```

Where "nnn.nnn.nnn.nnn" is the IP address and "NODE1\_SCS" the OpenVMS `SCSNODE` name of the remote node. See the HP DECnet-Plus documentation set for more information on how to maintain TCP/IP domain databases.

## 8.1.2 Some SQL Dialect-required Warnings not Delivered

Bugs 3651847 and 4532451

The required warnings (information codes) for such things as rows eliminated for nulls (`%RDB-I-ELIM_NULL`) and string truncation (`%RDB-I-TRUN_RTRV`) are not being returned for singleton `SELECT` and singleton `UPDATE` statements (statements that return a single row using the `INTO` clause). To demonstrate with a `PERSONNEL` database, use the following interactive SQL commands:

```
SQL> set dialect 'sql92';
SQL> attach 'filename sql$database';
SQL>
SQL> ! Force a row to contain NULL for SALARY_AMOUNT
SQL> update salary_history
cont> set salary_amount = NULL
cont> where employee_id = '00471'
cont> and salary_end = date vms'20-Aug-1981';
1 row updated
SQL>
SQL> declare :avg_sal integer(2);
SQL>
SQL> ! No informational generated (but is expected)
SQL> select avg(salary_amount) into :avg_sal
cont> from salary_history where employee_id = '00471'
cont> and salary_end >= date vms'1-AUG-1970';
SQL> show sqlca
SQLCA:
      SQLCAID:          SQLCA          SQLCABC:          128
      SQLCODE:          0
      SQLERRD:          [0]: 0
```

```

[1]: 0
[2]: 1
[3]: 0
[4]: 0
[5]: 0
SQLWARN0: 0      SQLWARN1: 0      SQLWARN2: 0
SQLWARN3: 0      SQLWARN4: 0      SQLWARN5: 0
SQLWARN6: 0      SQLWARN7: 0
SQLSTATE: 00000
SQL> print :avg_sal;
      AVG_SAL
      60893.86
SQL>
SQL> ! Non singleton query returns correct informational
SQL> select avg(salary_amount)
cont> from salary_history where employee_id = '00471'
cont> and salary_end >= date vms'1-AUG-1970';

      6.089385714285714E+004
1 row selected
%RDB-I-ELIM_NULL, null value eliminated in set function
SQL> show sqlca
SQLCA:
      SQLCAID:      SQLCA      SQLCABC:      128
      SQLCODE:      1003
      SQLERRD:      [0]: 0
                  [1]: 0
                  [2]: 1
                  [3]: 0
                  [4]: 0
                  [5]: 0
      SQLWARN0: 0      SQLWARN1: 0      SQLWARN2: 0
      SQLWARN3: 0      SQLWARN4: 0      SQLWARN5: 0
      SQLWARN6: 0      SQLWARN7: 0
      SQLSTATE: 01003
%RDB-I-ELIM_NULL, null value eliminated in set function
SQL>
SQL> rollback;

```

Since there is a row in the SALARY\_HISTORY table with a NULL in SALARY\_AMOUNT, the set function AVG should report an informational message (and return a special warning level SQLSTATE/SQLCODE value).

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

## 8.1.3 Partitioned Index with Descending Column and Collating Sequence

Bug 2797443

A known problem exists in which a query can return wrong results (number of rows returned is incorrect). This can happen on a table that has a multi-column, partitioned index in which one of the columns is sorted in descending order and the column has an associated collating sequence.

The following example can be used to demonstrate the problem.

```

$ sql$

create database file mf_collating.rdb alloc 10
  collating sequence french french
  create storage area area1 alloc 10
  create storage area area2 alloc 10
  create storage area area3 alloc 10;

create table tabl (id tinyint, r3 char (3));

insert into tabl (id, r3) values (1, 'a');
1 row inserted
insert into tabl (id, r3) values (1, 'b');
1 row inserted
insert into tabl (id, r3) values (1, 'f');
1 row inserted

create index y3 on tabl (id asc, r3 desc)
  store using (id, r3)
  in area1 with limit of (1, 'k')
  in area2 with limit of (1, 'e')
  otherwise in area3 ;

commit;

set flags 'strategy';

! Here is a query that returns the correct rows using sequential rather
! than indexed access.

select id, r3 from tabl where id = 1 and r3 <= 'e'
  optimize for sequential access;
Conjunct      Get      Retrieval sequentially of relation TAB1
  ID   R3
    1   a
    1   b
2 rows selected

! Here is the same query without the sequential access restriction.
! Note in the query strategy that index Y3 is used for data retrieval.
! This query ought to (but does not) return the same set of rows as
! for the sequential access query.

select id, r3 from tabl where id = 1 and r3 <= 'e';
Leaf#01 FFirst TAB1 Card=3
  BgrNdx1 Y3 [2:1] Fan=16
0 rows selected

```

## 8.1.4 RDO IMPORT Does Not Support FORWARD\_REFERENCES Created by SQL EXPORT

Recent versions of SQL EXPORT have included support for new features as they are added to Oracle Rdb. In particular, SQL now generates forward references for routines to allow references in the metadata prior to those routines being created. No such enhancements will be made to RDO IMPORT.

Due to changes in the CREATE STORAGE MAP statement for this release, the RDO IMPORT command is no longer able to process interchange (.RBR) files created by SQL EXPORT due to forward references to new

storage mapping routines. See the Oracle Rdb 7.1.4 Release Notes, Enhancement Chapter for details.

---

Note

*The RDO IMPORT command has been deprecated since the release of Oracle Rdb V7.0. Oracle recommends that users change all scripts to use the SQL IMPORT command in the future.*

---

The following example shows the reported error:

```
$ RDO
IMPORT 'thresh_alter_sql' INTO 'thresh' DICTIONARY IS NOT USED.
%RDO-W-UNSIMPORT, RDO IMPORT does not support all Oracle Rdb features, please
use SQL IMPORT
Exported by Oracle Rdb V7.1-301 Import/Export utility
A component of Oracle Rdb SQL V7.1-301
.
.
.
IMPORTing STORAGE AREA: RDB$SYSTEM
IMPORTing STORAGE AREA: AREA1
IMPORTing STORAGE AREA: AREA2
IMPORTing STORAGE AREA: DEFAULT_AREA
%RDO-E-EXTRADATA, unexpected data at the end of the RBR file
```

With the current release, you can work around this problem in one of the following ways:

1. Change the command to execute the SQL IMPORT command. This is the recommended and long term solution to this problem.
2. Change the SQL EXPORT command to include the NO FORWARD\_REFERENCES clause. This will eliminate the definitions which currently cause errors in RDO IMPORT. However, this interchange file may then not contain sufficient information to fully import the database.
3. The RMU/LOAD command can also be used to extract the data for individual tables. You must use the /MATCH\_NAME qualifier for Load.

## 8.1.5 New Attributes Saved by RMU/LOAD Incompatible With Prior Versions

Bug 2676851

To improve the behavior of unloading views, Oracle Rdb Release 7.1.2 changed the way view columns were unloaded so that attributes for view computed columns, COMPUTED BY and AUTOMATIC columns were saved. These new attributes are not accepted by prior releases of Oracle Rdb.

The following example shows the reported error trying to load a file from V7.1.2 under V7.1.0.4.

```
%RMU-F-NOTUNLFIL, Input file was not created by RMU UNLOAD
%RMU-I-DATRECSTO, 0 data records stored.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 21-OCT-2003 16:34:54.20
```

You can workaroud this problem by using the /RECORD\_DEFINITION qualifier and specifying the

FORMAT=DELIMITED option. However, this technique does not support LIST OF BYTE VARYING column unloading.

## 8.1.6 RDMS-E-RTNSBC\_INITERR, Cannot init. external routine server site executor

Execution of an external function or procedure with server site binding may unexpectedly fail.

The following example shows this problem.

```
%RDB-E-EXTFUN_FAIL, external routine failed to compile or execute successfully
-RDMS-E-EXTABORT, routine NNNNNNNNNN execution has been aborted
-RDMS-E-RTNSBC_INITERR, Cannot init. external routine server site executor;
reason XX
```

In this example, NNNNNNNNNN is the function name and XX is a decimal value such as 41.

While such errors are possible they are very unlikely to be seen, especially on systems that have had Rdb successfully installed. These errors usually indicate a problem with the environment. For instance, ensure that images RDMXSMvv.EXE, RDMXSRvv.EXE and RDMXSMPvv.EXE (where vv is the Rdb version) are installed and have the correct protections, as in the following example.

Directory DISK\$:<SYS6.SYSCOMMON.SYSLIB>

```
RDMXSM70.EXE;3          183    8-APR-2004 09:37:31.36  (RWED,RWED,RWED,RE)
RDMXSMP70.EXE;3        159    8-APR-2004 09:37:31.54  (RWED,RWED,RWED,RE)
RDMXSR70.EXE;3         67     8-APR-2004 09:37:31.74  (RWED,RWED,RWED,RE)
```

Total of 3 files, 409 blocks.

```
DISK$:<SYS6.SYSCOMMON.SYSLIB>.EXE
  RDMXSM70;3      Open Hdr Shared          Lnkbl
DISK$:<SYS6.SYSCOMMON.SYSLIB>.EXE
  RDMXSMP70;3    Open Hdr Shared          Prot Lnkbl  Safe
DISK$:<SYS6.SYSCOMMON.SYSLIB>.EXE
  RDMXSR70;3     Open Hdr Shared          Lnkbl
```

## 8.1.7 SYSTEM-F-INSFMEM Fatal Error With SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED in Galaxy Environment

When using the GALAXY SUPPORT IS ENABLED feature in an OpenVMS Galaxy environment, a *%SYSTEM-F-INSFMEM*, insufficient dynamic memory error may be returned when mapping record caches or opening the database. One source of this problem specific to a Galaxy configuration is running out of Galaxy Shared Memory regions. For Galaxy systems, GLX\_SHM\_REG is the number of shared memory region structures configured into the Galaxy Management Database (GMDB).

While the default value (for OpenVMS versions through at least V7.3-1) of 64 regions might be adequate for some installations, sites using a larger number of databases or row caches when the SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED features are enabled may find the default insufficient.

If a `%SYSTEM-F-INSFMEM`, *insufficient dynamic memory* error is returned when mapping record caches or opening databases, Oracle Corporation recommends that you increase the `GLX_SHM_REG` parameter by 2 times the sum of the number of row caches and number of databases that might be accessed in the Galaxy at one time. As the Galaxy shared memory region structures are not very large, setting this parameter to a higher than required value does not consume a significant amount of physical memory. It also may avoid a later reboot of the Galaxy environment. This parameter must be set on all nodes in the Galaxy.

---

#### Galaxy Reboot Required

*Changing the `GLX_SHM_REG` system parameter requires that the OpenVMS Galaxy environment be booted from scratch. That is, all nodes in the Galaxy must be shut down and then the Galaxy reformed by starting each instance.*

---

## 8.1.8 Oracle Rdb and OpenVMS ODS-5 Volumes

The OpenVMS Version 7.2 release introduced an Extended File Specifications feature, 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" directory trees.

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.

Oracle does support Oracle Rdb database file components on ODS-5 volumes provided that all of these files and directories used by Oracle Rdb strictly follow the ODS-2 file and directory name conventions. In particular, all file names must be specified entirely in uppercase and "special" characters in file or directory names are forbidden.

## 8.1.9 Optimization of Check Constraints

Bug 1448422

When phrasing constraints using the "CHECK" syntax, a poorer strategy can be chosen by the optimizer than when the same or similar constraint is phrased using referential integrity (PRIMARY and FOREIGN KEY) constraints.

For example, I have two tables T1 and T2, both with one column, and I wish to ensure that all values in table T1 exist in T2. Both tables have an index on the referenced field. I could use a PRIMARY KEY constraint on

T2 and a FOREIGN KEY constraint on T1.

```
SQL> alter table t2
cont>   alter column f2 primary key not deferrable;
SQL> alter table t1
cont>   alter column f1 references t2 not deferrable;
```

When deleting from the PRIMARY KEY table, Rdb will only check for rows in the FOREIGN KEY table where the FOREIGN KEY has the deleted value. This can be seen as an index lookup on T1 in the retrieval strategy.

```
SQL> delete from t2 where f2=1;
Get      Temporary relation      Retrieval by index of relation T2
      Index name  I2 [1:1]
Index only retrieval of relation T1
      Index name  I1 [1:1]
%RDB-E-INTEG_FAIL, violation of constraint T1_FOREIGN1 caused operation to fail
```

The failure of the constraint is not important. What is important is that Rdb efficiently detects that only those rows in T1 with the same values as the deleted row in T2 can be affected.

It is necessary sometimes to define this type of relationship using CHECK constraints. This could be necessary because the presence of NULL values in the table T2 precludes the definition of a primary key on that table. This could be done with a CHECK constraint of the form:

```
SQL> alter table t1
cont>   alter column f1
cont>   check (f1 in (select * from t2)) not deferrable;
SQL> delete from t2 where f2=1;
Get      Temporary relation      Retrieval by index of relation T2
      Index name  I2 [1:1]
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation T1
      Index name  I1 [0:0]
  Cross block entry 2
    Conjunct      Aggregate-F1      Conjunct
    Index only retrieval of relation T2
      Index name  I2 [0:0]
%RDB-E-INTEG_FAIL, violation of constraint T1_CHECK1 caused operation to fail
```

The cross block is for the constraint evaluation. This retrieval strategy indicates that to evaluate the constraint, the entire index on table T1 is being scanned and for each key, the entire index in table T2 is being scanned. The behavior can be improved somewhat by using an equality join condition in the select clause of the constraint:

```
SQL> alter table t1
cont>   alter column f1
cont>   check (f1 in (select * from t2 where f2=f1))
cont>     not deferrable;
```

or:

```
SQL> alter table t1
cont>   alter column f1
cont>   check (f1=(select * from t2 where f2=f1))
cont>     not deferrable;
```

In both cases the retrieval strategy will look like this:

```
SQL> delete from t2 where f2=1;
Get      Temporary relation      Retrieval by index of relation T2
  Index name  I2 [1:1]
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation T1
      Index name  I1 [0:0]
  Cross block entry 2
    Conjunct      Aggregate-F1      Conjunct
    Index only retrieval of relation T2
      Index name  I2 [1:1]
%RDB-E-INTEG_FAIL, violation of constraint T1_CHECK1 caused operation to fail
```

While the entire T1 index is scanned, at least the value from T1 is used to perform an index lookup on T2.

These restrictions result from semantic differences in the behavior of the "IN" and "EXISTS" operators with respect to null handling, and the complexity of dealing with non-equality join conditions.

To improve the performance of this type of integrity check on larger tables, it is possible to use a series of triggers to perform the constraint check. The following triggers perform a similar check to the constraints above.

```
SQL> create trigger t1_insert
cont> after insert on t1
cont> when (not exists (select * from t2 where f2=f1))
cont> (error) for each row;
SQL> create trigger t1_update
cont> after update on t1
cont> when (not exists (select * from t2 where f2=f1))
cont> (error) for each row;
SQL> ! A delete trigger is not needed on T1.
SQL> create trigger t2_delete
cont> before delete on t2
cont> when (exists (select * from t1 where f1=f2))
cont> (error) for each row;
SQL> create trigger t2_modify
cont> after update on t2
cont> referencing old as t2o new as t2n
cont> when (exists (select * from t1 where f1=t2o.f2))
cont> (error) for each row;
SQL> ! An insert trigger is not needed on T2.
```

The strategy for a delete on T2 is now:

```
SQL> delete from t2 where f2=1;
Aggregate-F1      Index only retrieval of relation T1
  Index name  I1 [1:1]
Temporary relation      Get      Retrieval by index of relation T2
  Index name  I2 [1:1]
%RDB-E-TRIG_INV_UPD, invalid update; encountered error condition defined for
trigger
-RDMS-E-TRIG_ERROR, trigger T2_DELETE forced an error
```

The trigger strategy is the index only retrieval displayed first. You will note that the index on T1 is used to examine only those rows that may be affected by the delete.

Care must be taken when using this workaround as there are semantic differences in the operation of the triggers, the use of "IN" and "EXISTS", and the use of referential integrity constraints.

This workaround is useful where the form of the constraint is more complex, and cannot be phrased using referential integrity constraints. For example, if the application is such that the value in table T1 may be spaces or NULL to indicate the absence of a value, the above triggers could easily be modified to allow for these semantics.

## 8.1.10 Using Databases from Releases Earlier Than V6.0

You cannot convert or restore databases earlier than V6.0 directly to V7.1. The RMU Convert command for V7.1 supports conversions from V6.0 through V7.0 only. If you have a V3.0 through V5.1 database, you must convert it to at least V6.0 and then convert it to V7.1. For example, if you have a V4.2 database, convert it first to at least V6.0, then convert the resulting database to V7.1.

If you attempt to convert a database created prior to V6.0 directly to V7.1, Oracle RMU generates an error.

## 8.1.11 Carryover Locks and NOWAIT Transaction Clarification

In NOWAIT transactions, the BLAST (Blocking AST) 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 carryover locks. There can be a delay before the transactions with carryover locks detect the presence of the NOWAIT transaction and release their carryover locks. You can detect this condition by examining the stall messages. If the "Waiting for NOWAIT signal (CW)" stall message appears frequently, the application is probably experiencing a decrease in performance, and you should consider disabling the carryover lock behavior.

## 8.1.12 Unexpected Results Occur During Read-Only Transactions on a Hot Standby Database

When using Hot Standby, it is typical to use the standby database for reporting, simple queries, and other read-only transactions. If you are performing these types of read-only transactions on a standby database, be sure you can tolerate a READ COMMIT level of isolation. This is because the Hot Standby database might be updated by another transaction before the read-only transaction finishes, and the data retrieved might not be what you expected.

Because Hot Standby does not write to the snapshot files, the isolation level achieved on the standby database for any read-only transaction is a READ COMMITTED transaction. This means that nonrepeatable reads and phantom reads are allowed during the read-only transaction:

- Nonrepeatable read operations: Allows the return of different results within a single transaction when an SQL operation reads the same row in a table twice. Nonrepeatable reads can occur when another transaction modifies and commits a change to the row between transactions. Because the standby

database will update the data when it confirms a transaction has been committed, it is very possible to see an SQL operation on a standby database return different results.

- Phantom read operations: Allows the return of different results within a single transaction when an SQL operation retrieves a range of data values (or similar data existence check) twice. Phantoms can occur if another transaction inserted a new record and committed the insertion between executions of the range retrieval. Again, because the standby database may do this, phantom reads are possible.

Thus, you cannot rely on any data read from the standby database to remain unchanged. Be sure your read-only transactions can tolerate a READ COMMIT level of isolation before you implement procedures that read and use data from a standby database.

## 8.1.13 Both Application and Oracle Rdb Using SYS\$HIBER

In application processes that use Oracle Rdb and the \$HIBER system service (possibly through RTL routines such as LIB\$WAIT), the application must ensure 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) feature is enabled.

The use of the \$WAKE system service by Oracle Rdb 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 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 BEGIN
      LIB$SIGNAL (STAT)
    END
    ! Hibernate. When the $HIBER completes, check to make
    ! sure that TIMER_FLAG is set indicating that the wait
    ! has finished.
    WHILE TIMER_FLAG = FALSE
    DO BEGIN
      SYS$HIBER()
    END
  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 BEGIN
```

```
LIB$SIGNAL (STAT)
END
END
```

The LIB\$K\_NOWAKE flag can be specified when using the OpenVMS LIB\$WAIT routine 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.

## 8.1.14 Bugcheck Dump Files with Exceptions at COSI\_CHF\_SIGNAL

In certain situations, Oracle Rdb bugcheck dump files 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 is needed when working with Oracle Rdb Worldwide Support.

## 8.1.15 Read-only Transactions Fetch AIP Pages Too Often

Oracle Rdb read-only transactions fetch Area Inventory Pages (AIP) to ensure that the logical area has not been modified by an exclusive read-write transaction. This check is needed because an exclusive read-write transaction does not write snapshot pages and these pages may be needed by the read-only transaction.

Because AIPs are always stored in the RDB\$SYSTEM area, reading the AIP pages could represent a significant amount of I/O to the RDB\$SYSTEM area for some applications. Setting the RDB\$SYSTEM area to read-only can avoid this problem, but it also prevents other online operations that might be required by the application so it is not a viable workaround in all cases.

This problem has been reduced in Oracle Rdb release 7.0. The AIP entries are now read once and then are not read again unless they need to be. This optimization requires that the carry-over locks feature be enabled (this is the default setting). If carry over locks are not enabled, this optimization is not enabled and the behavior is the same as in previous releases.

## 8.1.16 Row Cache Not Allowed While Hot Standby Replication is Active

The row cache feature may not be enabled on a hot standby database while replication is active. The hot standby feature will not start if row cache is enabled.

This restriction exists because rows in the row cache are accessed via logical dbkeys. However, information transferred to the standby database via the after image journal facility only contains physical dbkeys. Because there is no way to maintain rows in the cache via the hot standby processing, the row cache must be disabled when the standby database is open and replication is active.

A new command qualifier, `ROW_CACHE=DISABLED`, has been added to the `RMU Open` command. To open the hot standby database prior to starting replication, use the `ROW_CACHE=DISABLED` qualifier on the `RMU Open` command.

## 8.1.17 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. 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 a 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` shareable image. `SQL IMPORT` and `RMU Load` operations do, however, 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` does not directly consider the processes 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` 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 the new pages are faulted in. Once the sort operation completes and `SORT` returns back to Oracle Rdb, 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. Since that 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 for Oracle Rdb sort operations. When the operation can not be done in the available memory, SORT uses temporary disk files to hold the data as it is being sorted. The Oracle Rdb7 Guide to Database Performance and Tuning contains more detailed information about sort work files.

The logical name RDMS\$BIND\_SORT\_WORKFILES specifies how many work files sort 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 ten logical names, SORTWORK0 through SORTWORK9.

Normally, SORT places work files in the your SYS\$SCRATCH directory. By default, SYS\$SCRATCH 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 your work files reside on separate disks permits overlap of the SORT read/write cycle. You may also encounter cases where insufficient space exists on the SYS\$SCRATCH disk device (for example, while 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 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 fails 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 allocation within Oracle Rdb.

### 8.1.18 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 used carefully and sparingly.

The logical names are:

#### *Table 8-1 Sort Memory Logicals*

Logical	Definition
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 vary 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 processes 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.

## 8.1.19 The Halloween Problem

When a cursor is processing rows selected from a table, it is possible that another separate query can interfere with the retrieval of the cursor by modifying the index columns key values used by the cursor.

For instance, if a cursor selects all EMPLOYEES with LAST\_NAME >= 'M', it is likely that the query will use the sorted index on LAST\_NAME to retrieve the rows for the cursor. If an update occurs during the processing of the cursor which changes the LAST\_NAME of an employee from "Mason" to "Rickard", then it is possible that that employee row will be processed twice. First when it is fetched with name "Mason", and then later when it is accessed by the new name "Rickard".

The Halloween problem is a well known problem in relational databases. Access strategies which optimize the I/O requirements, such as Index Retrieval, can be subject to this problem. Interference from queries by other sessions are avoided by locking and are controlled by the ISOLATION LEVEL options in SQL, or the CONCURRENCY/CONSISTENCY options in RDO/RDML.

Oracle Rdb avoids this problem if it knows that the cursors subject table will be updated. For example, if the SQL syntax UPDATE ... WHERE CURRENT OF is used to perform updates of target rows, or the RDO/RDML MODIFY statement uses the context variable for the stream. Then the optimizer will choose an alternate access strategy if an update can occur which may cause the Halloween problem. This can be seen in the access strategy in Example 2-2 as a "Temporary relation" being created to hold the result of the cursor query.

When you use interactive or dynamic SQL, the UPDATE ... WHERE CURRENT OF or DELETE ... WHERE CURRENT OF statements will not be seen until after the cursor is declared and opened. In these environments, you must use the FOR UPDATE clause to specify that columns selected by the cursor will be updated during cursor processing. This is an indication to the Rdb optimizer so that it protects against the Halloween problem in this case. This is shown in Example 2-1 and Example 2-2.

The following example shows that the EMP\_LAST\_NAME index is used for retrieval. Any update performed will possibly be subject to the Halloween problem.

```

SQL> set flags 'strategy';
SQL> declare emp cursor for
cont> select * from employees where last_name >= 'M'
cont> order by last_name;
SQL> open emp;
Conjunct      Get      Retrieval by index of relation EMPLOYEES
  Index name  EMP_LAST_NAME [1:0]
SQL> close emp;

```

The following example shows that the query specifies that the column LAST\_NAME will be updated by some later query. Now the optimizer protects the EMP\_LAST\_NAME index used for retrieval by using a "Temporary Relation" to hold the query result set. Any update performed on LAST\_NAME will now avoid the Halloween problem.

```

SQL> set flags 'strategy';
SQL> declare emp2 cursor for
cont> select * from employees where last_name >= 'M'
cont> order by last_name
cont> for update of last_name;
SQL> open emp2;
Temporary relation      Conjunct      Get
Retrieval by index of relation EMPLOYEES
  Index name  EMP_LAST_NAME [1:0]
SQL> close emp2;

```

When you use the SQL precompiler, or the SQL module language compiler it can be determined from usage that the cursor context will possibly be updated during the processing of the cursor because all cursor related statements are present within the module. This is also true for the RDML/RDBPRE precompilers when you use the DECLARE\_STREAM and START\_STREAM statements and use the same stream context to perform all MODIFY and ERASE statements.

The point to note here is that the protection takes place during the open of the SQL cursor (or RDO stream), not during the subsequent UPDATE or DELETE.

If you execute a separate UPDATE query which modifies rows being fetched from the cursor then the actual rows fetched will depend upon the access strategy chosen by the Rdb optimizer. As the query is separate from the cursors query (i.e. doesn't reference the cursor context), then the optimizer does not know that the cursor selected rows are potentially updated and so cannot perform the normal protection against the Halloween problem.

## 8.2 SQL Known Problems and Restrictions

This section describes known problems and restrictions for the SQL interface for release 7.1.

### 8.2.1 Interchange File (RBR) Created by Oracle Rdb Release 7.1 Not Compatible With Previous Releases

To support the large number of new database attributes and objects, the protocol used by SQL EXPORT and SQL IMPORT has been enhanced to support more protocol types. Therefore, this format of the Oracle Rdb release 7.1 interchange files can no longer be read by older versions of Oracle Rdb.

Oracle Rdb continues to provide upward compatibility for interchange files generated by older versions.

Oracle Rdb has never supported backward compatibility, however, it was sometimes possible to use an interchange file with an older version of IMPORT. However, this protocol change will no longer permit this usage.

### 8.2.2 System Relation Change for International Database Users

Due to an error in creating the RDB\$FIELD\_VERSIONS system relation, another system relation, RDB\$STORAGE\_MAP\_AREAS, cannot be accessed if the session character sets are not set to DEC\_MCS.

This problem prevents the new Oracle Rdb GUIs, specifically the Oracle Rdb Schema Manager, from viewing indexes and storage maps from existing Oracle Rdb databases.

The problem can be easily corrected by executing the following SQL statement after attaching to the database:

```
SQL> UPDATE RDB$FIELD_VERSIONS SET RDB$FIELD_SUB_TYPE = 32767  
cont> WHERE RDB$FIELD_NAME = 'RDB$AREA_NAME';
```

### 8.2.3 Single Statement LOCK TABLE is Not Supported for SQL Module Language and SQL Precompiler

The new LOCK TABLE statement is not currently supported as a single statement within the module language or embedded SQL language compiler.

Instead you must enclose the statement in a compound statement. That is, use BEGIN... END around the statement as shown in the following example. This format provides all the syntax and flexibility of LOCK TABLE.

This restriction does not apply to interactive or dynamic SQL.

The following extract from the module language listing file shows the reported error if you use LOCK TABLE as a single statement procedure. The other procedure in the same module is acceptable because it uses a compound statement that contains the LOCK TABLE statement.

```

1 MODULE sample_test
2 LANGUAGE C
3 PARAMETER COLONS
4
5 DECLARE ALIAS FILENAME 'mf_personnel'
6
7 PROCEDURE a (SQLCODE);
8 LOCK TABLE employees FOR EXCLUSIVE WRITE MODE;
%SQL-F-WISH_LIST, (1) Feature not yet implemented - LOCK TABLE requires compound
statement
9
10 PROCEDURE b (SQLCODE);
11 BEGIN
12 LOCK TABLE employees FOR EXCLUSIVE WRITE MODE;
13 END;

```

To workaroud this problem of using LOCK TABLE for SQL module language or embedded SQL application, use a compound statement in an EXEC SQL statement.

## 8.2.4 Multistatement or Stored Procedures May Cause Hangs

Long-running multistatement or stored procedures can cause other users in the database to hang if the procedures obtain resources needed by those other users. Some resources obtained by the execution of a multistatement or stored procedure are not released until the multistatement or stored procedure finishes. Thus, any-long running multistatement or stored procedure can cause other processes to hang. This problem can be encountered even if the statement contains SQL 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 hangs forever.

Session 1:

```

SQL> attach 'filename MF_PERSONNEL';
SQL> create function LIB$WAIT (in real by reference)
cont> returns integer;
cont> external name LIB$WAIT location 'SYS$SHARE:LIBRTL.EXE'
cont> language general general parameter style variant;
SQL> commit;
.
.
.
$ SQL
SQL> attach 'filename 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 = LIBWAIT (5.0);
cont> set transaction read only;
cont> end loop;
cont> end;

```

Session 2:

## Oracle® Rdb for OpenVMS

```
$ RMU/BACKUP/LOG/ONLINE MF_PERSONNEL MF_PERSONNEL
```

From a third session, you can see that the backup process is waiting for a lock held in the first session:

```
$ RMU/SHOW LOCKS /MODE=BLOCKING MF_PERSONNEL
```

```
.  
. .  
Resource: nowait signal
```

ProcessID	Process Name	Lock ID	System ID	Requested	Granted
20204383	RMU BACKUP.....	5600A476	00010001	CW	NL
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 are released.

### 8.2.5 Use of Oracle Rdb from Shareable Images

If code in the image initialization routine of a shareable image makes any calls into Oracle Rdb, through SQL or any other means, access violations or other unexpected behavior may occur if Oracle Rdb images have not had a chance to do their own initialization.

To avoid this problem, applications must take one of the following steps:

- Do not make Oracle Rdb calls from the initialization routines of shareable images.
- Link in such a way that the RDBSHR.EXE image initializes first. You can do this by placing the reference to RDBSHR.EXE and any other Oracle Rdb shareable images last in the linker options file.

This is not a bug; it is a restriction resulting from the way OpenVMS image activation works.

## 8.3 Oracle RMU Known Problems and Restrictions

This section describes known problems and restrictions for the RMU interface for release 7.1.

### 8.3.1 RMU/BACKUP MAX\_FILE\_SIZE Option Has Been Disabled

The MAX\_FILE\_SIZE option of the RMU/BACKUP/DISK\_FILE qualifier for backup to multiple disk files has been temporarily disabled since it creates corrupt RBF files if the maximum file size in megabytes is exceeded and a new RBF file is created. It also does not give a unique name to the new RBF file but creates an RBF file with the same name but a new version number in the same disk directory. This will cause an RMU-F-BACFILCOR error on the restore and the restore will not complete.

The multi-file disk backup and restore will succeed if this option is not used. If this option is specified, a warning message is now output that this qualifier will be ignored.

The following example shows that the MAX\_FILE\_SIZE option, when used with the /DISK\_FILE qualifier on an RMU/BACKUP, will be ignored and a warning message will be output.

```
$ RMU/BACKUP /ONLINE          -
                          /NOCRC          -
                          /NOLOG          -
                          /NOINCREMENTAL  -
                          /QUIET_POINT    -
                          TEST_DB_DIR:TEST_DB
-
BACKUP_DIR_1:TEST_DB/DISK_FILE=(WRITER_THREADS=3,MAX_FILE_SIZE=10) ,-
BACKUP_DIR_2:/DISK_FILE=(WRITER_THREADS=3,MAX_FILE_SIZE=10) ,-
BACKUP_DIR_3:/DISK_FILE=(WRITER_THREADS=3,MAX_FILE_SIZE=10)

%RMU-W-DISABLEDOPTION, The MAX_FILE_SIZE option is temporarily disabled
and will be ignored
```

As a workaround to avoid this problem, do not specify the MAX\_FILE\_SIZE option with the /DISK\_FILE qualifier.

### 8.3.2 RMU Convert Fails When Maximum Relation ID is Exceeded

If, when relation IDs are assigned to new system tables during an RMU Convert of an Oracle Rdb V7.0 database to a V7.1 database, the maximum relation ID of 8192 allowed by Oracle Rdb is exceeded, the fatal error %RMU-F-RELMAXIDBAD is displayed and the database is rolled back to V7.0. Contact your Oracle support representative if you get this error. Note that when the database is rolled back, the fatal error %RMU-F-CVTROLSUC is displayed to indicate that the rollback was successful but caused by the detection of a fatal error and not requested by the user.

This condition only occurs if there are an extremely large number of tables defined in the database or if a large number of tables were defined but have subsequently been deleted.

The following example shows both the %RMU-F-RELMAXIDBAD error message if the allowed database relation ID maximum of 8192 is exceeded and the %RMU-F-CVROLSUC error message when the database has been rolled back to V7.0 since it cannot be converted to V7.1:

```
$rmu/convert mf_personnel
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.1-00
Are you satisfied with your backup of
  DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 and your backup of
  any associated .aij files [N]? Y
%RMU-I-LOGCONVRT, database root converted to current structure level
%RMU-F-RELMAXIDBAD, ROLLING BACK CONVERSION - Relation ID exceeds maximum
  8192 for system table RDB$RELATIONS
%RMU-F-CVROLSUC, CONVERT rolled-back for
  DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 to version V7.0
```

The following example shows the normal case when the maximum allowed relation ID is not exceeded:

```
$rmu/convert mf_personnel
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.1-00
Are you satisfied with your backup of
  DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 and your backup of
  any associated .aij files [N]? Y
%RMU-I-LOGCONVRT, database root converted to current structure level
%RMU-S-CVTDBSUC, database DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
  successfully converted from version V7.0 to V7.1
%RMU-I-CVTCOMSUC, CONVERT committed for
  DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 to version V7.1
```

### 8.3.3 RMU Unload /After\_Journal Requires Accurate AIP Logical Area Information

The RMU Unload /After\_Journal command uses the on-disk area inventory pages (AIPs) to determine the appropriate type of each logical area when reconstructing logical dbkeys for records stored in mixed-format storage areas. However, the logical area type information in the AIP is generally unknown for logical areas created prior to Oracle Rdb release 7.0.1. If the RMU Unload /After\_Journal command cannot determine the logical area type for one or more AIP entries, a warning message is displayed for each such area and may ultimately return logical dbkeys with a 0 (zero) area number for records stored in mixed-format storage areas.

In order to update the on-disk logical area type in the AIP, the RMU Repair utility must be used. The INITIALIZE=LAREA\_PARAMETERS=optionfile qualifier option file can be used with the TYPE qualifier. For example, to repair the EMPLOYEES table of the MF\_PERSONNEL database, you would create an options file that contains the following line:

```
EMPLOYEES /TYPE=TABLE
```

For partitioned logical areas, the AREA=name qualifier can be used to identify the specific storage areas that are to be updated. For example, to repair the EMPLOYEES table of the MF\_PERSONNEL database for the EMPID\_OVER storage area only, you would create an options file that contains the following line:

```
EMPLOYEES /AREA=EMPID_OVER /TYPE=TABLE
```

The TYPE qualifier specifies the type of a logical area. The following keywords are allowed:

- **TABLE**  
Specifies that the logical area is a data table. This would be a table created using the SQL CREATE TABLE syntax.
- **B-TREE**  
Specifies that the logical area is a B-tree index. This would be an index created using the SQL CREATE INDEX TYPE IS SORTED syntax.
- **HASH**  
Specifies that the logical area is a hash index. This would be an index created using the SQL CREATE INDEX TYPE IS HASHED syntax.
- **SYSTEM**  
Specifies that the logical area is a system record that is used to identify hash buckets. Users cannot explicitly create these types of logical areas.

---

Note

*This type should NOT be used for the RDB\$SYSTEM logical areas. This type does NOT identify system relations.*

---

- **BLOB**  
Specifies that the logical area is a BLOB repository.

There is no explicit error checking of the type specified for a logical area. However, an incorrect type may cause the RMU Unload /After\_Journal command to be unable to correctly return valid, logical dbkeys.

### 8.3.4 Do Not Use HYPERSORT with RMU Optimize After\_Journal Command

The OpenVMS Alpha V7.1 operating system introduced the high-performance Sort/Merge utility (also known as HYPERSORT). This utility takes advantage of the OpenVMS 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.

Because of this, 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.

### 8.3.5 Changes in EXCLUDE and INCLUDE Qualifiers for RMU Backup

The RMU Backup command no longer accepts both the Include and Exclude qualifiers in the same command. This change removes the confusion over exactly what gets backed up when Include and Exclude are specified on the same line, but does not diminish the capabilities of the RMU Backup command.

To explicitly exclude some storage areas from a backup, use the Exclude qualifier to name the storage areas to be excluded. This causes all storage areas to be backed up except for those named by the Exclude qualifier.

Similarly, the Include qualifier causes only those storage areas named by the qualifier to be backed up. Any storage area not named by the Include qualifier is not backed up. The Noread\_only and Noworm qualifiers continue to cause read-only storage areas and WORM storage areas to be omitted from the backup even if these areas are explicitly listed by the Include qualifier.

Another related change is in the behavior of EXCLUDE=\*. In previous versions, EXCLUDE=\* caused all storage areas to be backed up. Beginning with V7.1, EXCLUDE=\* causes only a root backup to be done. A backup created by using EXCLUDE=\* can be used only by the RMU Restore Only\_Root command.

### 8.3.6 RMU Backup Operations Should Use Only One Type of Tape Drive

When using more than one tape drive for an RMU Backup command, all of the tape drives must be of the same type (for example, all the tape drives must be TA90s or TZ87s or TK50s). Using different tape drive types (for example, 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 of 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 valid. This may be the case, for example, when using a TA90 and a TA90E.

Oracle Corporation recommends that, on a regular basis, you test your backup and recovery procedures and environment using a test system. You should restore the database and then recover using AIJs to simulate failure recovery of the production system.

Consult the Oracle Rdb7 Guide to Database Maintenance, the Oracle Rdb7 Guide to Database Design and Definition, and the Oracle RMU Reference Manual for additional information about Oracle Rdb backup and restore operations.

### 8.3.7 RMU/VERIFY Reports PGSPAMENT or PGSPMCLST Errors

RMU/VERIFY may sometimes report PGSPAMENT or PGSPMCLST errors when verifying storage areas. These errors indicate that the Space Area Management (SPAM) page fullness threshold for a particular data page does not match the actual space usage on the data page. For a further discussion of SPAM pages, consult the Oracle Rdb7 Guide to Database Maintenance.

In general, these errors will not cause any adverse affect on the operation of the database. There is potential for space on the data page to not be totally utilized, or for a small amount of extra I/O to be expended when searching for space in which to store new rows. But unless there are many of these errors then the impact should be negligible.

It is possible for these inconsistencies to be introduced by errors in Oracle Rdb. When those cases are discovered, Oracle Rdb is corrected to prevent the introduction of the inconsistencies. It is also possible for these errors to be introduced during the normal operation of Oracle Rdb. The following scenario can leave the SPAM pages inconsistent:

1. A process inserts a row on a page, and updates the threshold entry on the corresponding SPAM page to reflect the new space utilization of the data page. The data page and SPAM pages are not flushed to disk.
2. Another process notifies the first process that it would like to access the SPAM page being held by the process. The first process flushes the SPAM page changes to disk and releases the page. Note that it has not flushed the data page.
3. The first process then terminates abnormally (for example, from the DCL STOP/IDENTIFICATION command). Since that process never flushed the data page to disk, it never wrote the changes to the Recovery Unit Journal (RUJ) file. Since there were no changes in the RUJ file for that data page then the Database Recovery (DBR) process did not need to roll back any changes to the page. The SPAM page retains the threshold update change made above even though the data page was never flushed to disk.

While it would be possible to create mechanisms to ensure that SPAM pages do not become out of synch with their corresponding data pages, the performance impact would not be trivial. Since these errors are relatively rare and the impact is not significant, then the introduction of these errors is considered to be part of the normal operation of Oracle Rdb. If it can be proven that the errors are not due to the scenario above, then Oracle Product Support should be contacted.

PGSPAMENT and PGSPMCLST errors may be corrected by doing any one of the following operations:

- Recreate the database by performing:
  1. SQL EXPORT
  2. SQL DROP DATABASE
  3. SQL IMPORT
- Recreate the database by performing:
  1. RMU/BACKUP
  2. SQL DROP DATABASE
  3. RMU/RESTORE
- Repair the SPAM pages by using the RMU/REPAIR command. Note that the RMU/REPAIR command does not write its changes to an after-image journal (AIJ) file. Therefore, Oracle recommends that a full database backup be performed immediately after using the RMU/REPAIR command.

## 8.4 Known Problems and Restrictions in All Interfaces for Release 7.0 and Earlier

The following problems and restrictions from release 7.0 and earlier still exist.

### 8.4.1 Converting Single-File Databases

Because of a substantial increase in the database root file information for V7.0, you should ensure that you have adequate disk space before you use the RMU Convert command with single-file databases and V7.0 or higher.

The size of the database root file of any given database increases 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.

### 8.4.2 Row Caches and Exclusive Access

If a table has a row-level cache defined for it, the Row Cache Server (RCS) may acquire a shared lock on the table and prevent any other user from acquiring a Protective or Exclusive lock on that table.

### 8.4.3 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:

- ◆ Reserve the table for SHARED WRITE
- ◆ Close the database and disable row cache for the duration of the exclusive transaction
- ◆ 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.

### 8.4.4 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 Rdb scans extra partitions. A boundary value is a value specified in the WITH LIMIT OF clause. The following example, executed while the logical name RDMS\$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.

## 8.4.5 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 less than the smallest existing page size and the database has been manually opened or users are active, the add operation fails with one of the following errors:

```

%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

```

or

```

SQL>alter data file foo add storage area area6 page size 1 blocks;
%RDMS-F-NOEUACCESS, unable to acquire exclusive access to database

```

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, any recovery scenario fails. 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.

## 8.4.6 Support for Single-File Databases to Be Dropped in a Future Release

Oracle Rdb currently supports both single-file and multifile databases on all platforms. 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 Rdb recommends that users with single-file databases perform the following actions:

- ◆ Use the Oracle RMU commands, such as Backup and Restore, to make copies, backup, 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.

## 8.4.7 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 is not 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.

## 8.4.8 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 catch up to the application and are not able to process database pages that are logically ahead of the application in the RDB\$CHANGES system relation. 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 relation 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 V4.0 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.

## 8.5 SQL Known Problems and Restrictions for Oracle Rdb Release 7.0 and Earlier

The following problems and restrictions from Oracle Rdb Release 7.0 and earlier still exist.

### 8.5.1 SQL Does Not Display Storage Map Definition After Cascading Delete of Storage Area

When you drop 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> -- 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
```

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.

### 8.5.2 ARITH\_EXCEPT or Incorrect Results Using LIKE IGNORE CASE

When you use LIKE...IGNORE CASE, programs linked under Oracle Rdb V4.2 and V5.1, 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 (V6.0 or higher.)

## 8.5.3 Different Methods of Limiting Returned Rows from Queries

You can establish the query governor for rows returned from a query by using either the SQL SET QUERY LIMIT statement or a logical name. This note describes the differences between the two mechanisms.

If you define the RDMS\$BIND\_QG\_REC\_LIMIT logical name to a small value, the query often fails with no rows returned regardless of the value assigned to the logical. 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 relations 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 relation) is sufficient to read each column definition.

To see an indication of the queries executed against the system relations, define the RDMS\$DEBUG\_FLAGS logical name 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 RDM\$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 relations as part of query processing.

## 8.5.4 Suggestions for Optimal Use of 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 relation 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 the NUMBER OF BUFFERS IS clause in SQL to change the number of buffers).
- ◆ To distribute the disk I/O load, store the storage areas for the indexes on separate disk drives. Note that using the same storage area for multiple indexes results 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 Rdb7 Guide to Database 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 ten parallel process index creations (Index1, Index2, ... Index10) and one process with ten 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 ten 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 ten of the indexes serially.

*Table 8–2 Elapsed Time for Index Creations*

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
All10	00:03:26.66

## 8.5.5 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>     language SQL
cont>
```

## Oracle® Rdb for OpenVMS

```
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 Rdb.

### 8.5.6 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 stale 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 `RDMS$BIND_HOLD_CURSOR_SNAP` to the value 1 to force all hold cursors to fetch the result set into a cached data area. (The cached data area appears as a "Temporary Relation" in the optimizer strategy displayed by the `SET FLAGS 'STRATEGY'` statement or the `RDMS$DEBUG_FLAGS "S"` flag.) This logical name helps to stabilize the cursor to some degree.

## 8.5.7 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.

[| Contents](#)