

Oracle® Rdb for OpenVMS

Table of Contents

<u>Oracle® Rdb for OpenVMS</u>	1
<u>Release Notes</u>	2
<u>November 2019</u>	3
<u>Contents</u>	4
<u>Preface</u>	5
<u>Purpose of This Manual</u>	6
<u>Intended Audience</u>	7
<u>Access to Oracle Support</u>	8
<u>Document Structure</u>	9
<u>Chapter 1 Installing Oracle Rdb Release 7.3.3.2</u>	10
<u>1.1 Oracle Rdb on HPE OpenVMS Industry Standard 64</u>	11
<u>1.2 Requirements</u>	12
<u>1.2.1 Ensure No Processes Have RDMSHRP Image Activated</u>	12
<u>1.3 Deprecated Features</u>	14
<u>1.3.1 Deprecated Support for the PL/I Compiler</u>	14
<u>1.4 Intel Itanium Processor 9700 "Kittson" Certified</u>	15
<u>1.5 Hardware Emulation Support</u>	16
<u>1.6 Maximum OpenVMS Version Check</u>	17
<u>1.7 Database Format Changed</u>	18
<u>1.8 Using Databases from Releases Earlier than V7.0</u>	19
<u>1.9 Invoking the VMSINSTAL Procedure</u>	20
<u>1.10 Stopping the Installation</u>	21
<u>1.11 After Installing Oracle Rdb</u>	22
<u>1.12 VMS\$MEM RESIDENT USER Rights Identifier Required</u>	23

Table of Contents

<u>1.13 Installation, Configuration, Migration, Upgrade Suggestions</u>	24
<u>Chapter 2 Software Errors Fixed in Oracle Rdb Release 7.3.3.2</u>	27
<u>2.1 Software Errors Fixed That Apply to All Interfaces</u>	28
<u>2.1.1 RCS Bugchecks in DIO\$LACB AIP ENT REFRESH After Drop Table or Index</u>	28
<u>2.1.2 Unexpected SYSTEM-F-ACCVIO in PSIINDEX2DIVIDESCRCR When Deleting Ranked Index Records</u>	28
<u>2.2 SQL Errors Fixed</u>	30
<u>2.2.1 Unexpected Routine Call Failure Reporting RDB-E-OBSOLETE METADATA</u>	30
<u>2.2.2 Unable to DECLARE a LIST CURSOR When Table Cursor Renames the Columns</u>	30
<u>2.2.3 Unexpected Value in SQLCHRONO SCALE When SOLDA2 is Reused</u>	31
<u>2.2.4 Possible Wrong Result Returned From Aggregate Subselect Clauses</u>	31
<u>2.2.5 Unexpected SYSTEM-F-ACCVIO When Processing Complex Query</u>	33
<u>2.2.6 Unexpected Bugcheck When Trigger Executes in Called Stored Procedure</u>	33
<u>2.3 RMU Errors Fixed</u>	35
<u>2.3.1 Unexpected Ordering of Fields in Extract Records From RMU Extract</u>	35
<u>2.3.2 Unexpected Corrupted Files From NFS (TCPIP) Mounted Device</u>	36
<u>2.3.3 Unexpected Data Type of Fields in Extract Records From RMU Extract</u>	37
<u>Chapter 3 Software Errors Fixed in Oracle Rdb Release 7.3.3.1</u>	39
<u>3.1 Software Errors Fixed That Apply to All Interfaces</u>	40
<u>3.1.1 Query Slows Down Significantly Using Sequential Retrieval in Rdb Release 7.3</u>	40
<u>3.1.2 Ranked Index Bugcheck at PSIINDEX2JOINSCRCR</u>	42
<u>3.2 SQL Errors Fixed</u>	43
<u>3.2.1 Unexpected Action When CREATE Routine Was Missing LANGUAGE Clause</u>	43
<u>3.2.2 DECLARE DEFAULT TRANSACTION Does Not Work When Executed Prior to ATTACH</u>	43
<u>3.2.3 Unexpected Routine Call Failure Reporting RDB-E-OBSOLETE METADATA</u>	45
<u>3.2.4 Unexpected RDB-E-INVALID BLR When Defining a Constraint</u>	45
<u>3.2.5 Unexpected Results for Queries Using Query Rewrite of AND/OR Operators</u>	46
<u>3.2.6 Unexpected Sequence Creation by DECLARE LOCAL TEMPORARY TABLE Statement</u>	47
<u>3.2.7 Support for IDENTITY and Extended LIKE Table Clause for DECLARE LOCAL TEMPORARY TABLE</u>	49
<u>3.3 RMU Errors Fixed</u>	51
<u>3.3.1 Incorrect UNION Syntax Generated by RMU Extract for Some View Definitions</u>	51
<u>3.3.2 RMU Show User Now Reports if the Database is Opened for RESTRICTED Access</u>	52
<u>3.4 RMU Show Statistics Errors Fixed</u>	53
<u>3.4.1 RMU SHOW STATISTICS LOGICAL AREA STATISTICS Zoom Menu %SMG-F-INVROW Error</u>	53

Table of Contents

3.5 Oracle Trace Errors Fixed	58
3.5.1 Oracle TRACE Data Not Being Collected for All Database Connections.....	58
Chapter 4 Software Errors Fixed in Oracle Rdb Release 7.3.3.0	59
4.1 Software Errors Fixed That Apply to All Interfaces	60
4.1.1 Unexpected Bugcheck After an Exception is Reported Inserting LIST OF BYTE VARYING Column.....	60
4.1.2 Unexpected RDB-E-BAD DPB CONTENT Error During ATTACH to a Database.....	60
4.1.3 Possible Wrong Results From Partitioned Indices.....	61
4.1.4 Unexpected Results From Partitioned Index Query.....	61
4.1.5 Possible Lost Database Updates When Using RMU Backup Incremental.....	62
4.1.6 Deterministic Function Not Treated As Deterministic.....	63
4.1.7 Unexpected Bugcheck Generated When RDMS\$RUJ References Bad File Specification.....	64
4.1.8 Wrong Result From Aggregate Functions Using DISTINCT and FILTER Clauses.....	64
4.1.9 L\$EDIT Support Not Installed When Only DECSET License Was Found.....	66
4.1.10 Sub-optimal Performance Observed for COUNT Aggregate.....	66
4.1.11 Query Outline With MANDATORY Not Aborting Query When Compliance Not Possible...7	67
4.2 SQL Errors Fixed	68
4.2.1 Unexpected Favoring of the First Storage Area When Using FILL RANDOMLY Clause.....	68
4.2.2 VARIANCE Now Returns Zero for Single Row Groups for Oracle Dialects.....	68
4.2.3 SQL IMPORT DATABASE Does Not Restore THRESHOLDS for LIST Storage Map.....	69
4.2.4 Unexpected ACCVIO When Using INSERT ... RETURNING Statement.....	70
4.2.5 Unexpected RDB-F-REQ WRONG DB Error Reported by SHOW TRANSACTION.....	71
4.2.6 Unexpected RDB-E-SEONONEXT When Assigning DEFAULT to an IDENTITY Column.....	71
4.2.7 Unexpected Bugcheck When Executing Query With MIN, MAX or COUNT Functions.....	72
4.2.8 SQL Precompiler Not Setting Exit Status When INCLUDE File Missing.....	72
4.2.9 Unexpected Failure When WITH Clause Used With UNION Operator.....	73
4.2.10 Incorrectly Generated Routine Prototypes for Internal Calls in C++.....	74
4.2.11 Unexpected Value Inserted for Incorrect GUID Literal Value.....	74
4.2.12 Unexpected RDB-E-OBSOLETE METADA and RDMS-E-MODNEXTS Errors.....	75
4.2.13 Correction to Error Reporting of Database Not Yet Open.....	75
4.2.14 Unexpected Data Type Result From BITSTRING Function.....	76
4.3 RMU Errors Fixed	77
4.3.1 Handling RMU-F-AIJSEQAFT Error When Starting Continuous LogMiner.....	77
4.3.2 Unexpected RDMS-E-BAD CODE Reported When Using RMU Load to a LOCAL Temporary Table.....	78
4.3.3 RMU/BACKUP/AFTER IMAGE/FORMAT=NEW TAPE AIJ Sequencing Problem.....	78
4.3.4 Unexpected Bugcheck From RMU Verify Constraints.....	80
4.3.5 Unexpected Area Corruption After RMU Move Area of a UNIFORM Format Area.....	81
4.3.6 RMU Extract Generates an Extraneous Semicolon on DISABLE PRIMARY KEY Clause.....	82
4.3.7 RMU Dump Audit Always Assumed Type=ALL Even When Audit Classes Were Specified...82	82

Table of Contents

<u>4.4 LogMiner Errors Fixed</u>	84
<u>4.4.1 LogMiner Extracts Incorrect TSN Values From After Image Journal</u>	84
<u>4.5 RMU Show Statistics Errors Fixed</u>	85
<u>4.5.1 RMU SHOW STATISTICS Screen Did Not Display No Cluster Support Warning</u>	85
<u>4.6 Rdb SGA API Errors Fixed</u>	86
<u>4.6.1 RMUST CT LAPMS STATS Class Returned Incorrect RMUST T LOGICAL AREA ID Value</u>	86
<u>4.6.2 RMUST CT LAPMS STATS Class Returned Statistics for Unused Logical Areas</u>	86
<u>4.7 Oracle Trace Errors Fixed</u>	88
<u>4.7.1 EPC\$REGISTRAR Startup Access Violation if Unable to Access Rdb RUJ File</u>	88
<u>Chapter 5 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.2</u>	90
<u>5.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.2</u>	91
<u>5.1.1 New PCSI Support for Rdb Kit Installation and Deinstallation</u>	91
<u>5.1.2 Updated Support for RDMSS\$BIND CODE OPTIMIZATION</u>	92
<u>5.1.3 DUPLICATES ARE NOT ALLOWED Clause Added to ALTER INDEX Statement</u>	93
<u>5.1.4 New RMU/SET DATABASE Qualifier /ACCESS=[UN]RESTRICTED</u>	93
<u>5.1.5 New RMU/VERIFY Qualifier /VALIDATE for FLOAT Data Type Error Detection</u>	94
<u>5.1.6 New /ABORT Qualifier for the RMU/SET DATABASE Command</u>	95
<u>5.1.7 Smaller After Image Backup Files</u>	97
<u>Chapter 6 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.1</u>	98
<u>6.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.1</u>	99
<u>6.1.1 New and Modified Replication Network Error Handling Log Messages</u>	99
<u>6.1.2 RMU Show Statistics Feature To Select Screens To Include In STATISTICS.RPT</u>	100
<u>6.1.3 RMU BACKUP and RESTORE /OUTPUT Qualifier to Write Output to a File</u>	101
<u>6.1.4 AIJ Backup Data Compression Information is Now in the ABS Process Logs</u>	103
<u>6.1.5 RMU MOVE AREA and COPY DATABASE /OUTPUT Qualifier to Write Output to a File</u>	104
<u>6.1.6 Support for /OUTPUT=file-spec Qualifier Added to RMU Backup Plan File</u>	105
<u>6.1.7 New CREATE OR REPLACE Support for PROFILE</u>	106
<u>6.1.8 New CREATE OR REPLACE Support for VIEW</u>	107
<u>6.1.9 New CREATE OR REPLACE Support for SEQUENCE</u>	109
<u>6.1.10 New Options for SET LOGFILE Statement</u>	111
<u>6.1.11 New UNDECLARE CURSOR Statement</u>	112
<u>6.1.12 Enhanced LIKE Table Support in CREATE TABLE Statement</u>	112
<u>6.1.13 New TO_DSINTERVAL and TO_YMINTERVAL Functions</u>	115
<u>6.1.14 New CREATE DEFAULT AUDIT Statement</u>	117
<u>6.1.15 New ALTER DEFAULT AUDIT Statement</u>	121
<u>6.1.16 New DROP DEFAULT AUDIT Statement</u>	123
<u>6.1.17 New SESSION and GLOBAL Attributes for Sequences</u>	124
<u>6.1.18 New LogMiner Feature to Close and Immediately Reopen Table Output Files</u>	125

Table of Contents

<u>Chapter 7 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.0</u>	132
<u>7.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.0</u>	133
7.1.1 Relaxed Type Checking for DEFAULT Clause.....	133
7.1.2 New Statistics Screen Shows Top Processes Accessing a Table Logical Area.....	133
7.1.3 Relaxed Naming Rules for RMU Extract Option=MATCH Option.....	135
7.1.4 RMU/RESTORE Now Always Displays the %RMU-I-AIJRECFUL Message.....	136
7.1.5 New SQL Built-in Functions.....	138
7.1.5.1 New String Functions.....	138
7.1.5.2 New Aggregate Functions.....	140
7.1.6 -RMU-F-DBROOTFILE, -RMU-F-DBDATAFILE messages output with %RMU-F-BADAIJFILE.....	144
7.1.7 RMU Extract Now Outputs ALTER DATABASE For Storage Area Access Mode.....	145
7.1.8 RMU/RECOVER RMU-F-BACKUPNOAIJ, RMU-F-TSNNOSYNC, RMU-F-CANTSYNCTSNS Error Messages.....	145
7.1.9 Delimited Text Keywords Can Now Be Negated For RMU Load And Unload.....	148
7.1.10 RMU Load Now Supports User Defined Conversion Routines.....	149
7.1.11 New CARDINALITY Option for SHOW TABLE Command.....	151
7.1.12 New CONSTRAINT Naming for Domain Constraints.....	151
7.1.13 New AS Result-type Clause for CREATE SEQUENCE Statement.....	152
7.1.14 New GENERATED Column Support.....	153
7.1.15 Enhancements to INCLUDE Statement.....	154
7.1.16 New Support for DEFAULT Index NODE SIZE Calculation.....	155
7.1.17 New LANGUAGE Support From RMU Extract Command.....	156
7.1.18 Enhancements for CREATE and ALTER MODULE Statements.....	158
7.1.19 New RMU Dump Symbols Command.....	160
7.1.20 New Options to SET SOLDA Statement.....	161
7.1.21 More New Options to SET SOLDA Statement.....	162
<u>Chapter 8 Enhancements And Changes Provided in Oracle Rdb Release 7.3.2.1</u>	164
<u>8.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.2.1</u>	165
8.1.1 Oracle Rdb 7.3.2.1 Certified on OpenVMS 8.4-2 from VMS Software Inc. and Integrity i4 systems from HPE.....	165
8.1.2 RMU/SHOW AFTER JOURNAL [NO]CHECKPOINT Qualifier.....	165
8.1.3 Engine Error Logging.....	168
8.1.4 New MEDIAN Aggregate Function Added to SQL.....	169
8.1.5 New RMU/BACKUP/AFTER JOURNAL [NO]SPACE CHECK Qualifier.....	169
8.1.6 New Options to SET SOLDA Statement.....	171
8.1.7 New RMU Set Statistics Command.....	172
8.1.8 Multi-Aggregate Index Optimization.....	178
8.1.9 Use Old DPB Format for Rdb Change Database.....	179
8.1.10 LogMiner State Now in AIJ Options File. New RDM\$LOGMINER STATE Symbol.....	180
8.1.11 New /NO]MBX ASYNCH Qualifier for RMU/UNLOAD/AFTER.....	185
8.1.12 New /PAGE NUMBER Qualifier for RMU/DUMP and RMU/DUMP/BACKUP.....	186
8.1.13 New Information Table RDB\$SESSION PRIVILEGES Now Available.....	186

Table of Contents

<u>Chapter 9 Enhancements And Changes Provided in Oracle Rdb Release 7.3.2.0</u>	188
<u>9.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.2.0</u>	189
<u>9.1.1 New COMPRESSION OCTETS Clause for CREATE INDEX Statement</u>	189
<u>9.1.2 Enhancements for TRUNCATE TABLE Statement</u>	190
<u>9.1.3 RMU Extract Now Supports RECOMPILE Item</u>	192
<u>9.1.4 SQL Now Supports the MISSING VALUE Clause as Part of CREATE and ALTER DOMAIN Statement</u>	192
<u>9.1.5 Comma Statement Separator Now Deprecated</u>	193
<u>9.1.6 New Logical Name RDMS\$BIND DEADLOCK WAIT to Control Sub-second Deadlock Wait</u>	194
<u>9.1.7 Query Optimization Improvements for IN Clause</u>	194
<u>9.1.8 New SHOW AUDIT Command Added to Interactive SQL</u>	196
<u>9.1.9 RMU/RECLAIM Can Now Skip to the Next SPAM Interval and/or Storage Area to Avoid Lock Contention</u>	196
<u>9.1.10 RMU Open Statistics Supports PROCESS GLOBAL Qualifier</u>	199
<u>9.1.11 RMU/SHOW LOGICAL_NAME Now Supports /DESCRIPTION Qualifier</u>	199
<u>9.1.12 Using Per-Process Monitoring for RMU Show Statistics</u>	200
<u>9.1.12.1 Per-Process Monitoring Operational Modes</u>	201
<u>9.1.12.2 Per-Process Monitoring Facility Activation</u>	201
<u>9.1.12.3 Per-Process Monitoring Facility Process Activation</u>	203
<u>9.1.12.4 Per-Process Monitoring Run-Time Options</u>	204
<u>9.1.12.5 Detached Process Monitoring</u>	205
<u>9.1.12.6 Per-Process Monitoring Overview Information</u>	206
<u>9.1.13 RMU Error Messages Which Suggest Altering Backup File Attributes</u>	213
<u>9.1.14 Query Optimization Improvements for DATE ANSI Queries</u>	215
<u>9.1.15 New RMU Dump Metadata File Command</u>	216
<u>9.1.16 New REPLACE ROWS Qualifier Added to RMU Load Command</u>	217
<u>9.1.17 RMU/SET SHARED MEMORY/SECTION NAME</u>	218
<u>9.1.18 RESTART Clause of ALTER SEQUENCE No Longer Needs Value</u>	219
<u>9.1.19 New Options to SET SOLDA Statement</u>	220
<u>9.1.20 Support for External Authentication (LDAP)</u>	221
<u>9.1.21 New RMU Extract Options to Control Output for DATABASE and ALTER DATABASE Items</u>	221
<u>9.1.22 RMU/SET AFTER JOURNAL/SWITCH JOURNAL Now Can Create an Emergency AII</u>	223
<u>9.1.23 New Support for Second OpenVMS Account Password</u>	226
<u>9.1.24 New "Index Counts" Optimization for SORTED Indices</u>	226
<u>9.1.25 Support for Proxy Access to Remote Databases Using TCP/IP Transport</u>	227
<u>9.1.26 Support for INTEGER Result Type for COUNT Function</u>	228
<u>Chapter 10 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.3</u>	230
<u>10.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.3</u>	231
<u>10.1.1 Oracle Rdb 7.3.1.3 Certified on OpenVMS 8.4-1H1 from VMS Software Inc. and Integrity i4 systems from HPE</u>	231

Table of Contents

<u>Chapter 11 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.2</u>	232
<u>11.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.2</u>	233
<u>11.1.1 New FULBCKREQ Message Output When a Full Backup is Required</u>	233
<u>11.1.2 New TRACE Option for EXPORT DATABASE Statement</u>	234
<u>11.1.3 New /NOAFTER JOURNAL Qualifier to Disable AIJ File Creation by RMU/RECOVER</u>	235
<u>11.1.4 Enhance Dumper of Merge Range List</u>	236
<u>11.1.5 RMU Extract Now Extracts SYS GET DIAGNOSTIC Function</u>	237
<u>11.1.6 Alter Index Now Supports REVERSE and NOREVERSE Clauses</u>	237
<u>11.1.7 SQL Precompiler Now Generates C++ Compatible Intermediate C Source</u>	238
<u>11.1.8 New RMU/RECOVER Feature to Alter Storage Areas, After Journal and Record Cache Directories</u>	239
<u>11.1.9 RMU Unload Record Definition File Can Include Offset and Length Comment</u>	247
<u>11.1.10 New RMU/DUMP/BACKUP Enhanced Error Handling Features</u>	247
<u>11.1.11 New REVERSE Attribute for CREATE SEQUENCE Statement and IDENTITY Clause</u>	249
<u>Chapter 12 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.1</u>	251
<u>12.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.1</u>	252
<u>12.1.1 New LIMIT TO Qualifier Added to RMU Load Command</u>	252
<u>12.1.2 New BEFORE and SINCE Qualifiers Added to RMU Load Audit</u>	253
<u>12.1.3 New RMU/SHOW/STATISTICS Output File Periodic Buffer Flushes</u>	254
<u>12.1.4 New Error and Log Messages Added for Segmented String Verification</u>	256
<u>Chapter 13 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.0</u>	259
<u>13.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.0</u>	260
<u>13.1.1 Changes to Default and Limits Behavior in Oracle Rdb</u>	260
<u>13.1.2 New /ERROR LIMIT Qualifier Added as the Default to RMU/VERIFY</u>	262
<u>13.1.3 RMU/VERIFY Root Displays the Corrupt Page Table Entries</u>	264
<u>13.1.4 DECLARE LOCAL TEMPORARY TABLE Supports COMMENT IS Clause</u>	265
<u>13.1.5 Temporary Tables Now Support LARGE MEMORY Option</u>	265
<u>13.1.6 COUNT Now Returns BIGINT Result</u>	266
<u>13.1.7 Aggregate Functions Now Use BIGINT Counters</u>	266
<u>13.1.8 /[NO]KEY VALUES Qualifier Added to RMU/VERIFY/INDEX</u>	267
<u>13.1.9 The /LOCK TIMEOUT Qualifier Now Allows the Database Default</u>	268
<u>13.1.10 Compression of AIJ Backup Files for Automatic AIJ Backups</u>	269
<u>13.1.11 Global Statistics Sections for Better Performance</u>	269
<u>13.1.12 RMU/SET AUDIT Supports Wildcard Table and Column Names</u>	270
<u>13.1.13 RMU/BACKUP Database Root Verification Performance Enhancement</u>	272
<u>13.1.14 RMU/UNLOAD /AFTER JOURNAL New Qualifier /DELETES FIRST</u>	273
<u>13.1.15 Add Option to Pass Values to /CONFIRM During RESTORE Operation</u>	273
<u>13.1.16 Table Names Can Now Be Specified For Index Verification</u>	275
<u>13.1.17 New RMU/VERIFY Feature to Detect Orphan Hash Index Buckets</u>	276
<u>13.1.18 New COMPILE Clause for ALTER TRIGGER Statement</u>	277
<u>13.1.19 New COMPILE ALL TRIGGERS Clause for ALTER TABLE Statement</u>	278
<u>13.1.20 New RETRY Clause for ACCEPT Statement</u>	279

Table of Contents

13.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.0

<u>13.1.21 New Character Sets ISOLATIN2 and WIN LATIN2 Supported</u>	280
<u>13.1.22 Changes and Enhancements to Trigger Support</u>	280
<u>13.1.23 New RMU BACKUP RBF File BRH\$K ROOT1, BRH\$K ROOT2, BRH\$K ROOT3 Records /kroot records</u>	281
<u>13.1.24 New Functions NUMTODSINTERVAL, NUMTOYMINTERVAL Supported</u>	283
<u>13.1.25 RMU Dump Audit Command</u>	284
<u>13.1.26 New BIN TO NUM Numeric Function</u>	288
<u>13.1.27 RMU /PROGRESS REPORT and Control-T for RMU Backup and Restore</u>	289
<u>13.1.28 /[NO]SNAPSHOTS, /[NO]DATA FILE Added to RMU/MOVE AREA</u>	290
<u>13.1.29 Enhancements for Compression Support in SQL EXPORT DATABASE Command</u>	292
<u>13.1.30 /[NO]PARTITIONS Qualifier Added to RMU/ANALYZE/INDEXES</u>	293
<u>13.1.31 /[NO]PARTITIONS Qualifier Added to RMU/ANALYZE Storage Statistics</u>	297
<u>13.1.32 /[NO]PARTITIONS Qualifier Added to RMU/ANALYZE/PLACEMENT</u>	302
<u>13.1.33 New RMU/[NO]ASSIST Qualifier for Commands Using Tape Drives</u>	308
<u>13.1.34 New RMU/ALTER Feature to Modify the Area Header Root File Specification</u>	309
<u>13.1.35 Create Index Supports the REVERSE Keyword to Create Reverse Key Indices</u>	311
<u>13.1.36 Support for New Syntax for Sequence Generator Statements</u>	312
<u>13.1.37 RMU/SET AUDIT Supports MODULE, ROUTINE, SEQUENCE and VIEW</u>	313
<u>13.1.38 RMU/SHOW AUDIT Supports MODULE, ROUTINE, SEQUENCE and VIEW</u>	314
<u>13.1.39 SQL Now Supports SQL Standard Syntax for SET CONSTRAINTS ALL Statement</u>	315
<u>13.1.40 Support ANSI and ISO SQL Standard Length Units</u>	315
<u>13.1.41 New SET FLAGS Clause Supported by CREATE and ALTER PROFILE</u>	317
<u>13.1.42 New Support for SAVEPOINT Syntax and Semantics</u>	318
<u>13.1.42.1 SAVEPOINT Statement</u>	318
<u>13.1.42.2 RELEASE SAVEPOINT Statement</u>	320
<u>13.1.42.3 ROLLBACK TO SAVEPOINT Statement</u>	321
<u>13.1.43 New OPTIMIZE OUTLINE Clause Allows Outline Specification</u>	323
<u>13.1.44 RMU/DUMP/HEADER=ROW CACHE Now Displays Whether Row Cache is Enabled</u>	324
<u>13.1.45 RMU/LOAD Now Supports CSV Formatted Files</u>	329
<u>13.1.46 RMU/UNLOAD Now Supports CSV Formatted Files</u>	329
<u>13.1.47 RMU/UNLOAD Supports BITMAPPED SCAN Optimize Option</u>	330
<u>13.1.48 New EDIT STRING Clause for CREATE FUNCTION and CREATE MODULE Functions</u>	331
<u>13.1.49 Changes to RMU/VERIFY/CONSTRAINTS and ALTER TABLE Statement</u>	332
<u>13.1.50 New SORT Numeric Function</u>	333
<u>13.1.51 New MOD Numeric Function</u>	334
<u>13.1.52 New Data Types BINARY and BINARY VARYING</u>	335
<u>13.1.53 PERSONA SUPPORT is Enabled For All New Databases</u>	337
<u>13.1.54 New Dialects Support in SQL</u>	338
<u>13.1.55 New WITH Clause Provides Subquery Factoring</u>	338
<u>13.1.56 DECLARE LOCAL TEMPORARY VIEW Statement</u>	342
<u>13.1.57 Enhancements for Buffered Read Support in SQL EXPORT DATABASE Command</u>	344
<u>13.1.58 New BITMAPPED SCAN Clauses Added to OPTIMIZE Clause</u>	345
<u>13.1.59 New Support for Allocations Specified Using Quantified Numeric Literal</u>	346
<u>13.1.60 New SQL Functions Added</u>	347
<u>13.1.61 Optimized NOT NULL Constraint Execution</u>	347

Table of Contents

<u>13.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.0</u>	
<u>13.1.62 New RMU/LOAD Option CHARACTER_ENCODING XML</u>	348
<u>13.1.63 New MEMORY_ALLOCATION Clause for the GLOBAL_BUFFERS Definition</u>	349
<u>13.1.64 New REPLACE Statement</u>	349
<u>13.1.65 Query Optimization Improvements for IN Clause</u>	351
<u>Chapter 14 Documentation Corrections, Additions and Changes</u>	353
<u>14.1 Documentation Corrections</u>	354
<u>14.1.1 Oracle Rdb Release 7.3.x.x New Features Document Added</u>	354
<u>14.1.2 Undocumented /TRANSACTION=EXCLUSIVE Option for RMU Populate Cache</u>	354
<u>14.1.3 Action of DECLARE TRANSACTION When SQL\$DATABASE is Defined</u>	355
<u>14.1.4 RDB\$USAGE Field Values</u>	355
<u>14.1.5 Updated Documentation for RDM\$BIND_CODE_OPTIMIZATION Logical</u>	356
<u>14.1.6 New RDM\$BIND AIJ_BUFFER_POOL_COUNT Logical</u>	357
<u>14.1.7 RDM\$STT Image Optionally Installed</u>	358
<u>14.1.8 Clarification on the Affect of the RMU/OPEN/ACCESS=RESTRICTED Command</u>	359
<u>14.1.9 Some Optional System Tables Can Be Relocated Using a User Defined Storage Map</u>	360
<u>14.1.10 Logical Name RDM\$BIND_HOT_NETWORK_OBJECT Was Not Documented</u>	361
<u>14.1.11 Recovering an Oracle Rdb Database After RMU/RESTORE/ONLY_ROOT</u>	362
<u>14.1.12 Oracle Rdb Position on NFS Devices</u>	363
<u>14.1.13 RDM\$BIND STAREA_EMERGENCY_DIR Logical Name</u>	364
<u>14.1.14 RDMS-F-FULLAIJBKUP, Partially-Journalled Changes Made</u>	365
<u>14.1.15 Undocumented Hot Standby Logical Names</u>	368
<u>14.1.16 Missing Documentation for the TRANSACTION_TYPE Keyword for GET DIAGNOSTICS</u>	369
<u>14.1.17 Clarification on Using the RMU/UNLOAD TRIM=TRAILING Option</u>	370
<u>14.1.18 Corrections to the EDIT STRING Documentation</u>	372
<u>14.1.19 Changes and Improvements to the Rdb Optimizer and Query Compiler</u>	372
<u>14.1.20 Required Privileges for AUTHORIZATION Clause of CREATE MODULE</u>	375
<u>14.1.21 Sorting Capabilities in Oracle Rdb</u>	376
<u>14.1.22 RMU/VERIFY Process Quotas and Limits Clarification</u>	376
<u>14.1.23 RDM\$BIND MAX_DBR_COUNT Documentation Clarification</u>	377
<u>14.1.24 Database Server Process Priority Clarification</u>	377
<u>14.1.25 RDM\$BIND LOCK_TIMEOUT_INTERVAL Overrides the Database Parameter</u>	378
<u>14.1.26 Missing Tables Descriptions for the RDB\$EXPERT Collection Class</u>	379
<u>14.1.27 Missing Columns Descriptions for Tables in the Formatted Database</u>	379
<u>14.2 RDO, RDBPRE and RDB\$INTERPRET Features</u>	388
<u>14.2.1 New Request Options for RDO, RDBPRE and RDB\$INTERPRET</u>	388
<u>14.2.2 New Language Features for RDO and Rdb Precompiler</u>	390
<u>14.2.3 RDO Interface Now Supports Synonym References</u>	392
<u>14.3 Address and Phone Number Correction for Documentation</u>	393
<u>14.4 Online Document Format and Ordering Information</u>	394

Table of Contents

Chapter 15	Known Problems and Restrictions	395
15.1	Known Problems and Restrictions in All Interfaces	396
15.1.1	The Format of the RMU/ANALYZE/BINARY OUTPUT Files Can Change	396
15.1.2	RMU /VERIFY /KEY VALUES May Fail on Some Indices	397
15.1.3	REPLACE Statement Fails With Primary Key Constraint Failure When Used on a View	397
15.1.4	Possible Incorrect Results When Using Partitioned Descending Indexes	398
15.1.5	Remote Attach Stalls Before Detecting a Node is Unreachable	400
15.1.6	Application and Oracle Rdb Both Using SYSSHIBER	400
15.1.7	Unexpected RCS Termination	401
15.1.8	Changes for Processing Existence Logical Names	402
15.1.9	SQL Module or Program Fails with %SQL-F-IGNCASE BAD	403
15.1.10	External Routine Images Linked with PTHREAD\$RTL	403
15.1.11	Using Databases from Releases Earlier than V7.0	404
15.1.12	ILINK-E-INVOVRINI Error on I64	404
15.1.13	New Attributes Saved by RMU/UNLOAD Incompatible With Prior Versions	405
15.1.14	SYSTEM-F-INSFMEM Fatal Error With SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED in Galaxy Environment	405
15.1.15	Oracle Rdb and OpenVMS ODS-5 Volumes	406
15.1.16	Optimization of Check Constraints	406
15.1.17	Carryover Locks and NOWAIT Transaction Clarification	408
15.1.18	Unexpected Results Occur During Read-Only Transactions on a Hot Standby Database	409
15.1.19	Row Cache Not Allowed While Hot Standby Replication is Active	409
15.1.20	Excessive Process Page Faults and Other Performance Considerations During Oracle Rdb Sorts	409
15.1.21	Control of Sort Work Memory Allocation	411
15.1.22	The Halloween Problem	412
15.2	SQL Known Problems and Restrictions	414
15.2.1	Single Statement LOCK TABLE is Not Supported for SQL Module Language and SQL Precompiler	414
15.2.2	Multistatement or Stored Procedures May Cause Hangs	414
15.2.3	Use of Oracle Rdb from Shareable Images	415
15.3	Oracle RMU Known Problems and Restrictions	417
15.3.1	RMU/CONVERT Fails When Maximum Relation ID is Exceeded	417
15.3.2	RMU/UNLOAD/AFTER JOURNAL Requires Accurate AIP Logical Area Information	417
15.3.3	Do Not Use HYPERSORT with RMU/OPTIMIZE/AFTER JOURNAL Command	419
15.3.4	Changes in EXCLUDE and INCLUDE Qualifiers for RMU/BACKUP	419
15.3.5	RMU/BACKUP Operations Should Use Only One Type of Tape Drive	419
15.3.6	RMU/VERIFY Reports PGSPAMENT or PGSPMCLST Errors	420
15.4	Known Problems and Restrictions in All Interfaces for Release 7.0 and Earlier	422
15.4.1	Converting Single-File Databases	422
15.4.2	Row Caches and Exclusive Access	422
15.4.3	Exclusive Access Transactions May Deadlock with RCS Process	422
15.4.4	Strict Partitioning May Scan Extra Partitions	422
15.4.5	Restriction When Adding Storage Areas with Users Attached to Database	423

Table of Contents

<u>15.4 Known Problems and Restrictions in All Interfaces for Release 7.0 and Earlier</u>	
<u>15.4.6 Multiblock Page Writes May Require Restore Operation</u>	424
<u>15.4.7 Replication Option Copy Processes Do Not Process Database Pages Ahead of an Application</u>	424
<u>15.5 SQL Known Problems and Restrictions for Oracle Rdb Release 7.0 and Earlier</u>	425
<u>15.5.1 Different Methods of Limiting Returned Rows from Queries</u>	425
<u>15.5.2 Suggestions for Optimal Use of SHARED DATA DEFINITION Clause for Parallel Index Creation</u>	426
<u>15.5.3 Side Effect When Calling Stored Routines</u>	427
<u>15.5.4 Considerations When Using Holdable Cursors</u>	428
<u>15.5.5 AIJSERVER Privileges</u>	429

Oracle® Rdb for OpenVMS

Release Notes

Release 7.3.3.2

November 2019

Oracle Rdb Release Notes, Release 7.3.3.2 for OpenVMS

Copyright © 1984, 2019 Oracle Corporation. *All rights reserved.*

Primary Author: Rdb Engineering and Documentation group

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle, Java, Oracle Rdb, Hot Standby, LogMiner for Rdb, Oracle SQL/Services, Oracle CODASYL DBMS, Oracle RMU, Oracle CDD/Repository, Oracle Trace, and Rdb7 are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb Release 7.3.3.2. 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 7.3.3.2.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/us/support/contact/index.html> or visit <http://www.oracle.com/us/corporate/accessibility/support/index.html> if you are hearing impaired.

Document Structure

This manual consists of the following chapters:

Chapter 1	Describes how to install Oracle Rdb Release 7.3.3.2.
Chapter 2	Describes problems corrected in Oracle Rdb Release 7.3.3.2.
Chapter 3	Describes problems corrected in Oracle Rdb Release 7.3.3.1.
Chapter 4	Describes problems corrected in Oracle Rdb Release 7.3.3.0.
Chapter 5	Describes enhancements introduced in Oracle Rdb Release 7.3.3.2.
Chapter 6	Describes enhancements introduced in Oracle Rdb Release 7.3.3.1.
Chapter 7	Describes enhancements introduced in Oracle Rdb Release 7.3.3.0.
Chapter 8	Describes enhancements introduced in Oracle Rdb Release 7.3.2.1.
Chapter 9	Describes enhancements introduced in Oracle Rdb Release 7.3.2.0.
Chapter 10	Describes enhancements introduced in Oracle Rdb Release 7.3.1.3.
Chapter 11	Describes enhancements introduced in Oracle Rdb Release 7.3.1.2.
Chapter 12	Describes enhancements introduced in Oracle Rdb Release 7.3.1.1.
Chapter 13	Describes enhancements introduced in Oracle Rdb Release 7.3.1.0.
Chapter 14	Provides information not currently available in the Oracle Rdb documentation set.
Chapter 15	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb Release 7.3.3.2.

Chapter 1

Installing Oracle Rdb Release 7.3.3.2

This software update is installed using the OpenVMS VMSINSTAL utility.

NOTE

Oracle Rdb Release 7.3 kits are full kits. There is no requirement to install any prior release of Oracle Rdb when installing new Rdb Release 7.3 kits.

1.1 Oracle Rdb on HPE OpenVMS Industry Standard 64

The Oracle Rdb product family is available on the HPE OpenVMS Industry Standard 64 platform and the OpenVMS AlphaServer platform. In general, the functionality for one platform is available on the other platform. However, certain differences between the platforms may result in minor capability and functionality differences.

The database format for Oracle Rdb Release 7.3 is the same on both I64 and Alpha platforms and databases may be accessed simultaneously from both architectures in a cluster environment. Access to an Oracle Rdb Release 7.3 database from prior Rdb versions (on Alpha or VAX platforms) or from other systems on the network is available via the Oracle Rdb remote database server.

1.2 Requirements

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

- This Oracle Rdb release requires the following OpenVMS environments:
 - ◆ OpenVMS Alpha version 8.4 or later.
 - ◆ OpenVMS Industry Standard 64 version 8.4 or later.
- Some hardware configurations require the installation of OpenVMS V8.4–1H1 or later from VMS Software Inc. (VSI).
- Oracle strongly recommends that all available OpenVMS patches are installed on all systems prior to installing Oracle Rdb. Contact your HPE or VSI support representative for more information and assistance.
- Oracle Rdb must be shutdown before you install this update kit. That is, the command file SYS\$STARTUP:RMONSTOP73.COM should be executed before proceeding with this installation. If you have an OpenVMS cluster, you must shutdown the Rdb Release 7.3 monitor on all nodes in the cluster before proceeding.
- After executing RMONSTOP73.COM, no process on any system in the cluster should have any existing RDMSHRP73.EXE image activated. See [Section 1.2.1](#) for additional information.
- The installation requires approximately 280,000 blocks for OpenVMS Alpha systems.
- The installation requires approximately 500,000 blocks for OpenVMS I64 systems.

1.2.1 Ensure No Processes Have RDMSHRP Image Activated

The Oracle Rdb installation procedure checks to make sure that the Oracle Rdb Monitor (RDMMON) process is not running. However, it is also important to make sure that there are no processes on the cluster that share the system disk that have image activated a prior Rdb 7.3 version RDMSHRP image. Such processes may not be currently attached to a database but may do so in the future and could cause problems by using an older RDMSHRP image with a later Rdb installation.

The following command procedure can be used on each cluster node that shares the system disk to determine if there are any processes that have activated the RDMSHRP73.EXE image. This procedure should be executed by a privileged account after RMONSTOP73 has been run. Any processes that have RDMSHRP73.EXE activated at this point should be terminated prior to starting the Rdb installation procedure.

```
$ DEFINE /NOLOG /USER RDB$TMP 'RDB$TMP'
$ ANALYZE /SYSTEM
    SET OUTPUT RDB$TMP
    SHOW PROCESS /CHANNELS ALL
    EXIT
$ SEARCH /OUTPUT='RDB$TMP' 'RDB$TMP';-1 RDMSHRP73.EXE,"PID:"
$ SEARCH 'RDB$TMP' RDMSHRP73.EXE /WINDOW=(1,0)
$ DELETE /NOLOG 'RDB$TMP';*
```

In the following example, the process 2729F16D named "FOO\$SERVER" has the image RDMSHRP73.EXE activated even after RMONSTOP73.COM has been executed and this process is terminated prior to starting the Rdb installation procedure:

```
$ @SYS$STARTUP:RMONSTOP73.COM
```


Oracle® Rdb for OpenVMS

```
.  
. .  
$ @FIND_RDMSHRP73_PROC.COM  
  
OpenVMS system analyzer  
  
Process index: 016D   Name: FOO$SERVER   Extended PID: 2729F16D  
0240 7FEF4460 8384F300 $1$DGA2:[VMS$COMMON.SYSLIB]RDMSHRP73.EXE;222  
  
$ STOP/IDENTIFICATION=2729F16D
```

1.3 Deprecated Features

1.3.1 Deprecated Support for the PL/I Compiler

Oracle no longer supports the PL/I compiler on OpenVMS Alpha systems. Any existing support (SQL\$PRE/PLI qualifier, and LANGUAGE PLI in the SQL Module language) will be provided "AS IS". However, Oracle Rdb no longer plans to enhance or test this functionality.

1.4 Intel Itanium Processor 9700 "Kittson" Certified

For this release of Oracle Rdb on HPE Integrity servers, the Intel Itanium Processor 9700 series, code named "Kittson", is the newest processor for which Rdb is certified. Please note that OpenVMS V8.4-2L1 or later is required for this class of processors.

1.5 Hardware Emulation Support

Stromasys and AVTware offer Alpha emulator products that prolong the use of OpenVMS Alpha applications. These products include Charon-AXP from Stromasys and vtAlpha from AVTware. The Charon-AXP and vtAlpha products emulate complete Alpha systems allowing OpenVMS applications, layered products, tools, and middleware to run unmodified. For more details on these options, see our web page about these platforms:

<https://www.oracle.com/database/technologies/related/alpha-vax-emulators.html>

1.6 Maximum OpenVMS Version Check

OpenVMS Version 8.4–x is the maximum supported version of OpenVMS for this release of Oracle Rdb.

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.

1.7 Database Format Changed

The Oracle Rdb on-disk database format is 730 as shown in the following example.

```
$ RMU/DUMP/HEADER databasename
...
  Oracle Rdb structure level is 73.0
...
```

An RMU/CONVERT operation is required for databases created by or accessed by Oracle Rdb V7.0, V7.1 or V7.2 to be accessed with Rdb Release 7.3.

Prior to upgrading to Oracle Rdb Release 7.3 and prior to converting an existing database to Oracle Rdb Release 7.3 format, Oracle strongly recommends that you perform a full database verification (with the "RMU /VERIFY /ALL" command) along with a full database backup (with the "RMU /BACKUP" command) to ensure a valid and protected database copy.

1.8 Using Databases from Releases Earlier than V7.0

The RMU Convert command for Oracle Rdb V7.3 supports conversions from Rdb V7.0, V7.1 and V7.2 format databases only. This restriction also applies to the RMU Restore command which implicitly uses RMU Convert during the restore operation.

```
$ RMU/RESTORE/NOCD/DIRECTORY=SYS$DISK:[ ] [-]TESTING.RBF
%RMU-I-AIJRSTAVL, 0 after-image journals available for use
%RMU-I-AIJISOFF, after-image journaling has been disabled
%RMU-I-LOGCONVRT, database root converted to current structure level
%RMU-S-CVTDBSUC, database DEV1:[DOCUMENTATION.T]TESTING.RDB;1 successfully
converted from version V7.0 to V7.3
%RMU-I-CVTCOMSUC, CONVERT committed for DEV1:[DOCUMENTATION.T]TESTING.RDB;1
to version V7.3
%RMU-W-USERECCOM, Use the RMU Recover command. The journals are not available.
```

If you have an Oracle Rdb V3.0 through V6.1 format database or database backup, you must convert it to at least Oracle Rdb V7.0 format and then convert it to Oracle Rdb V7.3 format.

For example, if you have a V4.2 format database, you must convert it first to at least Oracle Rdb V7.0 format, then convert it to Oracle Rdb V7.3 format. This might be achieved by using Rdb V6.1 in an OpenVMS Alpha multiversion environment to convert from V4.2 to V6.1. Then installing Rdb V7.1 in that multiversion environment to convert V6.1 to V7.1. Then installing Rdb V7.3 in that multiversion environment to convert V7.1 to V7.3.

Note

This multi-step conversion will require you to use RMU/CONVERT/COMMIT so rolling back to Rdb V4.2 from V7.3 will not be possible. Oracle recommends that full database backups be performed prior to starting conversions to later versions.

Alternately, use the SQL EXPORT DATABASE statement to save the database structure and recreate it using the SQL IMPORT DATABASE statement. This allows a single step migration but will take more I/O and time because the database is completely rebuilt.

If you attempt to convert or restore a database format that is prior to Oracle Rdb V7.0 format directly to Oracle Rdb V7.3 format, Oracle RMU generates an error.

```
$ rmu/convert scratch/noconfirm
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.3-12 on OpenVMS Alpha V8.4
%RMU-F-CVRTUNS, The minimum database version that can be converted is version
70.
%RMU-F-FTL_CNV, Fatal error for CONVERT operation at 22-APR-2015 15:48:04.61
```

1.9 Invoking the VMSINSTAL Procedure

The installation procedure for Oracle Rdb has been simplified as compared with prior Oracle Rdb major releases. All Oracle Rdb components are always installed and the number of prompts during the installation has been reduced. The installation procedure is the same for Oracle Rdb for OpenVMS Alpha and Oracle Rdb for OpenVMS I64.

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

- To install the Oracle Rdb for OpenVMS I64 kit

```
@SYS$UPDATE:VMSINSTAL RDBV73320IM073 device-name
```

- To install the Oracle Rdb for OpenVMS Alpha kit

```
@SYS$UPDATE:VMSINSTAL RDBV73320AM073 device-name
```

device-name

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

1.10 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.

1.11 After Installing Oracle Rdb

This update provides a new Oracle TRACE facility definition for Oracle Rdb. 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, "RDBVMSV7.3".

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=RDBVMSV7.3 -  
_ $ /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.

1.12 VMS\$MEM_RESIDENT_USER Rights Identifier Required

Oracle Rdb requires 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.

1.13 Installation, Configuration, Migration, Upgrade Suggestions

Oracle Rdb Release 7.3 fully supports mixed-architecture clusters for AlphaServer systems and HPE Integrity servers.

In certain development environments, it may be helpful to incorporate a VAX system into the AlphaServer systems and HPE Integrity servers cluster. While HPE and Oracle believe that in most cases this will not cause problems to the computing environment, we have not tested it extensively enough to provide support. It is possible that VAX systems in a cluster may cause a problem with the cluster performance or stability. Should this happen, the VAX systems in the cluster which are causing the difficulty should be removed.

Oracle continues to support mixed architecture clusters of VAX systems and AlphaServer systems with direct database access using Rdb V7.0. Oracle Rdb V7.1 runs natively on Alpha systems and clusters. All Rdb versions include a built-in remote network database server allowing cross-architecture and cross-version application and database access.

All systems directly accessing the same database within a cluster environment must be running an identical version of Oracle Rdb (where the first 4 digits of the version number match). Access from other versions of Oracle Rdb may be accomplished with the built-in remote network database server for cross-version database access.

When moving applications from existing Alpha or VAX configurations to new environments containing Integrity Server systems, there are numerous possible paths depending on the requirements of individual sites. In general, this can be as straightforward as adding a new node to an already existing AlphaServer systems cluster or standalone system, except the node is an HPE Integrity server. [Table 1-1, Migration Suggestions](#), considers several possible situations and recommended steps to take.

Table 1-1 Migration Suggestions

Case	You Wish To...	You should...
1	Add an Integrity server to an existing cluster of Alpha servers	<ol style="list-style-type: none"> 1. Verify database(s) using RMU/VERIFY/ALL. 2. Backup database(s) using RMU/BACKUP. 3. Install Rdb 7.3 on Integrity and Alpha nodes. 4. Convert database(s) to the Rdb 7.3 structure level using RMU/CONVERT. 5. Verify database(s) again using RMU/VERIFY/ALL. 6. Backup database(s) using RMU/BACKUP. 7. Access database(s) from Alpha and Integrity directly by specifying database root file

		specification(s) in SQL ATTACH statements.
2	Add an Integrity server to an existing mixed cluster of VAX and Alpha nodes and access an Rdb database from all nodes. Disks used for the database are accessible from all nodes.	<ol style="list-style-type: none"> 1. Verify database(s) using RMU/VERIFY/ALL. 2. Backup database(s) using RMU/BACKUP. 3. Install Rdb 7.3 on Integrity and Alpha nodes. 4. Convert database(s) to the Rdb 7.3 structure level using RMU/CONVERT. 5. Verify database(s) again using RMU/VERIFY/ALL. 6. Backup database(s) using RMU/BACKUP. 7. Access database(s) from Alpha and Integrity nodes directly by specifying database root file specification(s) in SQL ATTACH statements. 8. Access the database from VAX node(s) using the Rdb built-in network server (remote database) by specifying one of the Alpha or Integrity node names in SQL ATTACH statements. 9. After thorough testing, remove VAX nodes from the cluster.
3	Move database(s) to new disks and add an Integrity server to an existing cluster.	<ol style="list-style-type: none"> 1. Use RMU/COPY with an options file to move the database files to the new disks. 2. Follow the steps for case 1 or case 2.
4	Continue to use Rdb primarily from VAX or Alpha nodes using earlier releases. Add an Integrity server for application testing purposes.	<ol style="list-style-type: none"> 1. Install Rdb 7.3 on Integrity node. 2. Access existing database(s) from Integrity node by specifying one of the Alpha or VAX node names in the SQL ATTACH statements. 3. When testing is complete, follow the steps in case 1 or case 2.
5	Add an Integrity server to an existing cluster of Alpha servers or create a new cluster from an existing stand-alone Alpha server by adding one or more new	<ol style="list-style-type: none"> 1. Verify database(s) using RMU/VERIFY/ALL.

	Integrity servers.	<ol style="list-style-type: none"> 2. Backup database(s) using RMU/BACKUP. 3. Install Rdb 7.3 on Integrity and Alpha nodes. 4. Convert database(s) to the Rdb 7.3 structure level using RMU/CONVERT. 5. Verify database(s) again using RMU/VERIFY/ALL. 6. Backup database(s) using RMU/BACKUP. 7. Access database(s) from Alpha and Integrity directly by specifying database root file specification in the SQL ATTACH statements.
6	Create a new stand-alone Integrity Server system or cluster of Integrity Servers and move database(s) to the new environment.	<ol style="list-style-type: none"> 1. Verify database(s) using RMU/VERIFY/ALL. 2. Install Rdb 7.3 on new system(s). 3. Back up database(s) on the existing cluster using RMU/BACKUP. 4. Copy backup file(s) to the new system (or, if using tape media, make the tapes available to the new system). 5. Restore database(s) on the new system using RMU/RESTORE specifying the location of each database file in an options file. 6. Verify the new database using RMU/VERIFY/ALL.

Refer to the Oracle Rdb documentation set for additional information and detailed instructions for using RMU and remote databases.

Note that database parameters might need to be altered in the case of accessing a database from a larger number of systems in a cluster.

Chapter 2

Software Errors Fixed in Oracle Rdb Release 7.3.3.2

This chapter describes software errors that are fixed by Oracle Rdb Release 7.3.3.2.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 RCS Bugchecks in DIO\$LACB_AIP_ENT_REFRESH After Drop Table or Index

Bugs 4738878, 20692223

It was possible for the Row Cache Server to fail with an exception similar to the following after a table or index (logical area) was dropped from the database.

```
COSI-F-BUGCHECK, internal consistency failure
Exception occurred at RDMRCS72\DIO$LACB_AIP_ENT_REFRESH + 00000670
Called from RDMRCS72\DIO$READY + 00000730
Called from RDMRCS72\DIOCCH$ALLOC_URCA + 00000A70
Called from RDMRCS72\DIOCCH$PREPARE_URCRVEC + 000001F0
```

The database would then shutdown. Any subsequent attempt to access the database would result in a database recovery process failing with the following error. Note that the logical area number would vary, depending on what area was deleted.

```
RDMS-F-CANTFINDLAREA, cannot locate logical area 66 in area inventory page list
Exception occurred at RDMDBR72\DIO$LACB_CREATE + 00000280
```

The RCS would fail after it scanned the cache creating a list of rows to flush to disk. If a table or index was dropped after the list was created but before the RCS began writing the entries to disk, the attempt to reference the now deleted logical area would return an unexpected error. The DBR would fail because it would try to apply updates to the now deleted logical area. Normally, a successful checkpoint would prevent the DBR from attempting to recover a table that was deleted. This problem is documented in My Oracle Support alert 455055.1.

The RCS will now ignore entries in its checkpoint list if the logical area is deleted.

This problem is less likely to be encountered if a manual database checkpoint is issued prior to dropping the table or index.

```
RMU /SERVER RECORD_CACHE CHECKPOINT root-file-spec
```

The only reliable way to avoid this problem is to disable row caching before dropping a table or index.

This problem has been corrected in Oracle Rdb Release 7.3.3.2.

2.1.2 Unexpected SYSTEM-F-ACCVIO in PSIINDEX2DIVIDESCRCR When Deleting Ranked Index Records

Bugs 15855499, 11656062 and 30457812

Oracle® Rdb for OpenVMS

In very rare cases, while processing the removal of entries from a ranked index, an error may occur in the re-adjustment of the index node layout leading to the following access violation:

```
%SYSTEM-F-ACCVIO, access violation, reason mask=04,  
virtual address=0000000003FBA000, PC=FFFFFFFF80002831, PS=00000009  
Saved PC = 0000000080A4FA80 : RDMSHRP73\PSIINDEX2DIVIDESC + 000002D0  
Saved PC = 00000000809E2360 : RDMSHRP73\PSII2REMOVEBOTTOM + 000016D0  
...
```

There are several workarounds to this problem. Once the affected index is found, it can be disabled by using `ALTER INDEX ... MAINTENANCE IS DISABLED`, or if the index resides in a `UNIFORM` storage area an `ALTER INDEX ... REBUILD ALL PARTITIONS` can be used. The latter command truncates the index (avoiding the `DELETE` operation that causes the error) and builds the index from the table data.

This problem has been corrected in Oracle Rdb Release 7.3.3.2.

2.2 SQL Errors Fixed

2.2.1 Unexpected Routine Call Failure Reporting RDB-E-OBSOLETE_METADATA

Bug 29610611

In Oracle Rdb Release V7.3.3, a change was made to correct an error calling SQL stored routines which executed transaction statements (SET TRANSACTION, START TRANSACTION, LOCK TABLE, COMMIT or ROLLBACK), used the GET DIAGNOSTICS statement (or the function SYS_GET_DIAGNOSTIC), or accessed declared local temporary tables and views.

These types of routines need the activating module name to be declared prior to executing the routine. This provides a required context for Oracle Rdb.

In some cases, applications built with the SQL Pre-compiler or the SQL Module Language compiler would fail with the following error.

```
%RDB-E-OBSOLETE_METADATA, request references metadata objects that no longer exist  
-RDMS-E-MODNEXTS, module <module_name> does not exist in this database
```

or this error

```
%RDB-E-OBSOLETE_METADATA, request references metadata objects that no longer exist  
-RDMS-F-BAD_SYM, unknown module symbol - <module_name>
```

Here <module_name> would be the name of the module containing the routine being called. This indicates that the module was not declared to Oracle Rdb prior to the call to the routine. This should be done automatically by SQL.

This problem has been corrected in Oracle Rdb Release 7.3.3.2.

2.2.2 Unable to DECLARE a LIST CURSOR When Table Cursor Renames the Columns

Bug 8941047

Once you rename a LIST OF BYTE VARYING column in the table cursor, you cannot declare a list cursor on that column against that table cursor.

The following example shows that neither the table column name nor the renamed column can be used.

```
SQL> declare jq3  
cont>     read only table cursor  
cont>     for select resume res from resumes  
cont>     ;  
SQL>  
SQL> declare jq4  
cont>     read only scroll list cursor
```

```

cont>      for select resume
cont>          where current of jq3;
%SQL-F-NOTINTBLCUR, Column, RESUME, not specified in table cursor, JQ3
SQL>
SQL> declare jq6
cont>      read only scroll list cursor
cont>      for select res
cont>          where current of jq3;
%SQL-F-FLDNOTCRS, Column RES was not found in the tables in current scope

```

This has now been fixed. The new name (created by the AS clause or by an implicit rename) can now be used to reference the table cursor's columns. In addition, SQL also allows the base column name to be referenced as long as this name is not ambiguous.

This problem has been corrected in Oracle Rdb Release 7.3.3.2.

2.2.3 Unexpected Value in SQLCHRONO_SCALE When SQLDA2 is Reused

Bug 30033543

In prior releases of Oracle Rdb, the SQLDA fields SQLCHRONO_PRECISION and SQLCHRONO_SCALE were not correctly reset when the SQLDA2 was reused for a new query. This is normally not a problem unless the new query also references a date/time type. In such cases, the SQLCHRONO_SCALE is used to identify the type of date/time value and this might cause the application to think that a DATE VMS column was a DATE ANSI, TIME or TIMESTAMP type. The SQLCHRONO_SCALE field should be set to SQLDA2_DT_DATE_VMS (0) for DATE VMS types.

This problem has been corrected in Oracle Rdb Release 7.3.3.2. Oracle Rdb now clears these fields in the SQLDA2. As a workaround, the application should set SQLCHRONO_PRECISION and SQLCHRONO_SCALE in the SQLVAR2 portions of the SQLDA2.

2.2.4 Possible Wrong Result Returned From Aggregate Subselect Clauses

In prior releases of Oracle Rdb, the SQL interface was ignoring some select-expression clauses during query compile. This may cause some subselects to return the wrong results. This only occurs if the subselect uses the OFFSET, FETCH FIRST (LIMIT TO), and ORDER BY clauses.

In a simple case, this elimination of the clauses is harmless as the ORDER BY cannot alter the result.

```

SQL> select
cont>      (select avg(salary_amount) from salary_history
cont>          where salary_end is null
cont>          order by employee_id) as average edit using 'SZ(9).99'
cont> from
cont>      rdb$database;
          AVERAGE
          31922.79
1 row selected
SQL>

```

However, by using the FETCH FIRST or OFFSET clause, the input rows to the aggregate are reduced and the result should be different.

The following example shows the problem; the result should not be the same as the previous query.

```
SQL> select
cont>     (select avg(salary_amount) from salary_history
cont>     where salary_end is null
cont>     offset 10 rows
cont>     order by employee_id
cont>     fetch first 10 rows only) as average edit using 'SZ(9).99'
cont> from
cont>     rdb$database;
Tables:
  0 = RDB$DATABASE
  1 = SALARY_HISTORY
Cross block of 2 entries  Q0
Cross block entry 1
  Aggregate: 0:AVG (1.SALARY_AMOUNT) Q2
  Conjunct: MISSING (1.SALARY_END)
  Get      Retrieval sequentially of relation 1:SALARY_HISTORY
Cross block entry 2
  Conjunct: 0.DBKEY = _DBKEY'8:611:0'
  Firstn: 1
  Get      Retrieval by DBK of relation 0:RDB$DATABASE
          AVERAGE
          31922.79
1 row selected
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.3.2.

This example shows the correct result. Note that the resulting strategy now reflects the available clauses.

```
SQL> select
cont>     (select avg(salary_amount) from salary_history
cont>     where salary_end is null
cont>     offset 10 rows
cont>     order by employee_id
cont>     fetch first 10 rows only) as average edit using 'SZ(9).99'
cont> from
cont>     rdb$database;
Tables:
  0 = RDB$DATABASE
  1 = SALARY_HISTORY
Cross block of 2 entries  Q0
Cross block entry 1
  Aggregate: 0:AVG (1.SALARY_AMOUNT) Q2
  Firstn: 10
  Skipn: 10
  Conjunct: MISSING (1.SALARY_END)
  Get      Retrieval by index of relation 1:SALARY_HISTORY
          Index name  SH_EMPLOYEE_ID [0:0]
Cross block entry 2
  Conjunct: 0.DBKEY = _DBKEY'8:611:0'
  Firstn: 1
  Get      Retrieval by DBK of relation 0:RDB$DATABASE
          AVERAGE
          21811.50
1 row selected
```

SQL>

Note

This change requires any SQL Module Language or SQL Precompiler sources to be recompiled using this version (or later) of Oracle Rdb. Interactive SQL and Dynamic SQL will implicitly process new queries correctly.

2.2.5 Unexpected SYSTEM-F-ACCVIO When Processing Complex Query

Bug 30382086

In rare cases, while processing a complex query, SQL might generate an SYSTEM-F-ACCVIO. Reformating the query might avoid this issue; such as eliminating the ORDER BY clause.

The following example shows an example of such a query.

```
SQL> select *
cont> from
cont>     (select *,sum(cnt)
cont>       from
cont>         (select *
cont>           from
cont>             (select a1,a2,(select i1 from t2 where a3=a1) as cnt
cont>               from t1
cont>               )
cont>             where cnt > 0
cont>           )
cont>         group by a1
cont>       )
cont> order by 2
cont> ;
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=0000000000000000, PC=FFFFFFFF804CC0D0, PS=0000001B
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.3.2.

2.2.6 Unexpected Bugcheck When Trigger Executes in Called Stored Procedure

Bug 30520710

In Oracle Rdb V7.3.3 and V7.3.3.1, it is possible to receive bugcheck dumps with this footprint.

- ***** Exception at 00000000813C7870 : RDMSHRP73\RDMS\$\$EXECUTE_ETRG + 000005D0
- %SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address= 000000000000
- Saved PC = 000000008141A4F0 : RDMSHRP73\RDMS\$\$EXE_ACTION + 0000A490
- Saved PC = 0000000081507020 : RDMSHRP73\RDMS\$\$C_EXE_ACTION + 00000070
- Saved PC = 0000000080CC2730 : RDMSHRP73\RDMS_EXE_INTERP + 0000FD10

Oracle® Rdb for OpenVMS

- Saved PC = 0000000815005F0 : RDMSHRP73\JSB_REQX + 00000220
- Saved PC = 000000081443610 : RDMSHRP73\RDMS\$TOP_START_SEND_RECEIVE + 00002E90

This problem can only occur when the routine being called is defined within a MODULE that has global variables. When the trigger gets fired by the top-most called procedure, it encounters an invalid state. The procedure is called as a single statement CALL; that is outside a compound statement.

This problem has been corrected in Oracle Rdb Release 7.3.3.2. The module-variable-initialization code now correctly preserves the call state.

2.3 RMU Errors Fixed

2.3.1 Unexpected Ordering of Fields in Extract Records From RMU Extract

Bug 29869005

In prior releases of Oracle Rdb, the RMU Extract command ignored the setting of the POSITION_COLUMN option when generating record definitions for COBOL, PASCAL, and C.

This meant that any table that was changed using the ALTER TABLE ... ALTER COLUMN ... AFTER COLUMN clause or the ALTER TABLE ... ALTER COLUMN ... BEFORE COLUMN clause were not displayed as expected.

The following example shows creation of the new column ordering.

```
SQL> create table SAMPLE
cont>     (a integer
cont>       ,c char(30)
cont>       ,t timestamp
cont>     );
SQL>
SQL> alter table SAMPLE
cont>     drop column C
cont> ;
SQL>
SQL> --> use AFTER COLUMN to relocate the column
SQL> alter table SAMPLE
cont>     add column C
cont>         char(40)
cont>     after column A
cont> ;
SQL>
SQL> show table (column) SAMPLE
Information for table SAMPLE

Columns for table SAMPLE:
Column Name          Data Type          Domain
-----
A                    INTEGER
C                    CHAR(40)
T                    TIMESTAMP(2)

SQL>
```

Here is the unexpected output from RMU Extract (new column ordering is not reflected in the output record typedef definition).

```
$ rmu/extract/item=table ABC /option=(noheader,match:SAMPLE)/language=CC

/* Table: SAMPLE (null terminated)
*/

#ifdef _SAMPLE_
```

```
#define _SAMPLE_
typedef struct {
    int A;
    SQL_TIMESTAMP T;
    char C[41];
} SAMPLE;
#endif // SAMPLE
```

This problem has been corrected in Oracle Rdb Release 7.3.3.2. RMU Extract now obeys the setting of the POSITION_COLUMN (or NOPOSITION_COLUMN) option as well as defaulting to POSITION_COLUMN when record definitions are extracted.

2.3.2 Unexpected Corrupted Files From NFS (TCPIP) Mounted Device

Bug 30100614

In a prior version of Oracle Rdb, a customer reported problems when attempting to use files created by **RMU Backup After Image** which were written directly to NFS (Network File System) disks. Such devices are mounted by TCPIP and appear as normal OpenVMS devices and are supported for use as output from RMU commands.

In some cases, the created backup files cannot be read by **RMU Recover** or **RMU Load After Image**. Oracle has determined that a probable flaw exists for the NFS TCPIP support in OpenVMS for Windows systems. Our testing of similarly configured TCPIP mounted NFS disks on Linux systems does not exhibit this same problem.

This problem only occurs when the /COMPRESSION qualifier is used either on the **RMU Backup After** command line or established by the **RMU Set After_Journal Backups** command.

The following example shows using the **RMU Backup After** command.

```
$ RMU/BACKUP-
  /AFTER-
  /LOG-
  /COMPRESS=ZLIB=1 -
  TEST$:TEST -
  DNFS4:[TEST_SYSTEM.ABS_COMP_NFS_7]TEST.BAIJ
$
```

The following example shows using the **RMU Set After_Journal** command.

```
$ RMU/SET AFTER -
  TEST$:TEST-
  /BACKUPS=(AUTOMATIC, -
            BACKUP_FILE=DNFS4:[TEST_SYSTEM.ABS_COMP_NFS_7]TEST.BAIJ, -
            COMPRESSION=ZLIB:1)-
  /LOG
$
```

This is the resulting error from RMU when using the corrupted file:

```
$ RMU-
  /RECOVER-
```


Oracle® Rdb for OpenVMS

```
/ROOT=TEST$:TEST-
/NOLOG-
/NOAUTOMATIC-
/NOCONFIRM-
DNFS4:[TEST_SYSTEM.ABS_COMP_NFS_7]TEST.BAIJ
%RMU-I-LOGRECDDB, recovering database file
DISK$9:[TEST_SYSTEM.DATABASE]TEST.RDB;2
%RMU-E-RECFALLED, fatal, unexpected roll-forward error detected at AIJ
record 29229
%RMU-I-RECLASTTSN, last successfully processed transaction was TSN 16236
%RMU-I-RECUNTIL, work-around: roll forward AIJ using
/UNTIL="27-AUG-2019 23:29:32.88" qualifier
%COSI-F-BUGCHECK, internal consistency failure
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-I-BUGCHKDMP, generating bugcheck dump file ...RMUBUGCHK.DMP;
%RMU-F-FTL_RCV, Fatal error for RECOVER operation at 27-AUG-2019 23:41:08.36
$
```

A simple DCL TYPE command can reveal that the file is corrupted.

```
$ TYPE-
/OUTPUT=NL:-
DNFS4:[TEST_SYSTEM.ABS_COMP_NFS_7]TEST.BAIJ
%TYPE-W-READERR, error reading DNFS4:[TEST_SYSTEM.ABS_COMP_NFS_7]TEST.BAIJ;1
-RMS-F-IRC, illegal record encountered; VBN or record number = 2126
```

As a workaround to this problem, Oracle recommends that the RMU Backup After_Image command with /COMPRESSION direct the output to an OpenVMS device and then use the DCL command BACKUP/DELETE/VERIFY to copy the RMU backup file to the NFS mounted device.

This problem has been corrected in Oracle Rdb Release 7.3.3.2. This release of Oracle Rdb avoids this problem by simplifying the I/O performed by RMU when using compression.

2.3.3 Unexpected Data Type of Fields in Extract Records From RMU Extract

Bug 29844726

In prior releases of Oracle Rdb, the RMU Extract command generated incorrect PICTURE definitions for scaled integers when /LANGUAGE=COBOL was used. The resulting definitions had maximum leading digits and n fractional places. The total digits caused COBOL to promote the field to a larger data type.

The following CREATE TABLE statement shows the definitions of example columns.

```
SQL> create table TEST_SCALED_INT
cont>      (FLD1 INTEGER
cont>      ,FLD2 INTEGER(2)
cont>      ,FLD3 BIGINT
cont>      ,FLD4 BIGINT(2)
cont>      );
SQL>
```

Here is the expected output from RMU Extract:

```
** Table: TEST_SCALED_INT
```

Oracle® Rdb for OpenVMS

```
01 TEST_SCALED_INT.  
   05 FLD1           picture S9(9) comp.  
   05 FLD2           picture S9(7)V9(2) comp.  
   05 FLD3           picture S9(18) comp.  
   05 FLD4           picture S9(16)V9(2) comp.
```

This problem has been corrected in Oracle Rdb Release 7.3.3.2. RMU Extract now correctly defines these PICTURE clauses. A note will be added to the documentation stating that some scaled integers from SQL cannot be represented in COBOL and may require editing of the generated record definitions.

Chapter 3

Software Errors Fixed in Oracle Rdb Release 7.3.3.1

This chapter describes software errors that are fixed by Oracle Rdb Release 7.3.3.1.

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 Query Slows Down Significantly Using Sequential Retrieval in Rdb Release 7.3

Bug 28604088

A customer observed that a query slowed down significantly using sequential retrieval after upgrading to Oracle Rdb 7.3 from Oracle Rdb 7.2.

The following shows the needed simple tables and query to reproduce the problem.

```
create database filename 'testdata';

create table T1 (
    JOIN_ID
        INTEGER,
    IDENT_ID
        INTEGER);

create table T2 (
    JOIN_ID
        INTEGER,
    IDENT_ID
        INTEGER);

create index T1_JOIN_NDX on T1 (JOIN_ID)
    type is SORTED
    node size 960
    percent fill 85;

create unique index T1_IDENT_NDX on T1 (IDENT_ID)
    type is SORTED
    node size 960
    percent fill 85;

create index T2_JOIN_NDX on T2 (JOIN_ID);

update rdb$relations
    set rdb$cardinality = 20000
    where rdb$relation_name = 'T1';

update rdb$relations
    set rdb$cardinality = 800000
    where rdb$relation_name = 'T2';

commit work;

! Now execute the following query
!
SELECT * from T1 t1 where ident_id IN (
select b.ident_id
from T1 b
    inner join T2 v on ( b.join_id = v.join_id )
```

```

where b.join_id = 123456 );
Tables:
  0 = T1
  1 = T1
  2 = T2
Conjunct: <agg0> <> 0
Match    (Agg Outer Join)  Q1
  Outer loop
  Match_Key:0.IDENT_ID
  Sort: 0.IDENT_ID(a)
  Get Retrieval sequentially of relation 0:T1
  Inner loop
  Match_Key:1.IDENT_ID
  Aggregate: 0:COUNT-ANY (<subselect>) Q2
  Cross block of 2 entries  Q2
  Cross block entry 1
    Leaf#01 Sorted 1:T1 Card=20000
    Bool: 1.JOIN_ID = 123456
    FgrNdx  T1_IDENT_NDX [0:0] Fan=46
    BgrNdx1 T1_JOIN_NDX [1:1] Fan=46
    Keys: 1.JOIN_ID = 123456
  Cross block entry 2
    Index only retrieval of relation 2:T2
    Index name  T2_JOIN_NDX [1:1]
    Keys: 1.JOIN_ID = 2.JOIN_ID
0 rows selected

```

Notice that it uses sequential retrieval on table 0:T1 instead of an index retrieval even though T1_IDENT_NDX is available for index retrieval.

The problem was caused by a change in the ordering of the sequence of indices in Oracle Rdb 7.3.

The workaround is to modify the query using the EXISTS clause to replace the IN clause.

```

SELECT * from T1 t1 where EXISTS (
select b.ident_id
  from T1 b
    inner join T2 v on ( b.join_id = v.join_id )
 where b.join_id = 123456 and
        b.ident_id = t1.ident_id);

```

```

Tables:
  0 = T1
  1 = T1
  2 = T2
Conjunct: <agg0> <> 0
Match  Q1
  Outer loop      (zig-zag)
  Match_Key:0.IDENT_ID
  Index_Key:IDENT_ID
  Get      Retrieval by index of relation 0:T1
  Index name  T1_IDENT_NDX [0:0]
  Inner loop
  Match_Key:1.IDENT_ID
  Aggregate: 0:COUNT-ANY (<subselect>) Q2
  Cross block of 2 entries  Q2
  Cross block entry 1
    Leaf#01 Sorted 1:T1 Card=20000
    Bool: 1.JOIN_ID = 123456
    FgrNdx  T1_IDENT_NDX [0:0] Fan=46
    BgrNdx1 T1_JOIN_NDX [1:1] Fan=46

```

Oracle® Rdb for OpenVMS

```
Keys: 1.JOIN_ID = 123456
Cross block entry 2
Index only retrieval of relation 2:T2
Index name T2_JOIN_NDX [1:1]
Keys: 1.JOIN_ID = 2.JOIN_ID
0 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.3.3.1.

3.1.2 Ranked Index Bugcheck at PSIINDEX2JOINSCR

Bug 28854908

Under very rare circumstances, it is possible that the insertion of a new record into a table may cause a bugcheck during the processing of the insertion of the associated key into a sorted ranked index.

```
***** Exception at 000000008100A3E0 : RDMSHRP731\PSIINDEX2JOINSCR + 00000310
%COSI-F-BUGCHECK, internal consistency failure
```

As sorted ranked indexes are used for internal database metadata storage, it is possible this problem may also be seen during the creation of database objects such as views and tables, but only if, over time, there has been a very large number of these objects created and subsequently deleted.

If this problem occurs, rebuilding the index may provide a workaround. Alternatively please contact Oracle Support for further assistance.

This problem has been corrected in Oracle Rdb Release 7.3.3.1.

3.2 SQL Errors Fixed

3.2.1 Unexpected Action When CREATE Routine Was Missing LANGUAGE Clause

In Oracle Rdb Release 7.3.3, the LANGUAGE clause is now optional and may be inherited from the module header. Unfortunately, standalone CREATE FUNCTION and CREATE PROCEDURE statements did not correctly check for a missing LANGUAGE clause. When this clause was omitted, SQL interpreted these statements as DECLARE FUNCTION or DECLARE PROCEDURE statements.

The following example shows that the CREATE didn't store the definition in the database.

```
SQL> create function LIB$LP_LINES () returns integer; external;
SQL> show function LIB$LP_LINES;
  No functions found
SQL>
```

The workaround for this problem is to include the LANGUAGE clause. Future versions of Oracle SQL will ensure that the LANGUAGE clause is present.

```
SQL> create function LIB$LP_LINES () returns integer; external;
%SQL-F-RTNNOTDEF, function or procedure LIB$LP_LINES is not defined
-SQL-E-NOLANGSPEC, No LANGUAGE specified for external routine
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.3.1.

3.2.2 DECLARE DEFAULT TRANSACTION Does Not Work When Executed Prior to ATTACH

Bug 6502369

In prior releases of Oracle Rdb, the DECLARE DEFAULT TRANSACTION statement did not work correctly when it was executed prior to the first database attach. It should be expected to supply a global default for the session in such cases; as does the DECLARE TRANSACTION statement.

The following example shows the problem. A user is assigned a PROFILE with a DEFAULT TRANSACTION which should be used once the attach occurs.

```
SQL> ! setup test database, profile, user
SQL> !
SQL> create database filename test1;
SQL> create profile RO_USER
cont>   default transaction read only;
SQL> create user J_JONES
cont>   identified externally profile ro_user;
SQL> commit;
SQL> show user J_JONES
  J_JONES
  Identified externally
```

Oracle® Rdb for OpenVMS

```
Account is unlocked
Profile: RO_USER
No roles have been granted to this user
SQL> show profile ro_user;
RO_USER
Default transaction read only
SQL> disconnect all;
SQL> !
SQL> ! declare default before attaching to any database
SQL> !
SQL> declare default transaction;
SQL> show transaction
Transaction information:
Statement constraint evaluation is DEFERRED (off)
SQL> !
SQL> attach 'filename test1';
SQL> show transaction;
Transaction information:
Statement constraint evaluation is DEFERRED (off)
```

```
On the default alias
Transaction characteristics:
Read Write
```

```
Transaction information returned by base system:
no transaction is in progress
- session ID number is 2
Session transaction modes (all)
Prestarted transactions are ENABLED for this session
Session user has a DEFAULT TRANSACTION
Snapshots are ENABLED IMMEDIATE
SQL> select current_user from rdb$database;
```

```
J_JONES
1 row selected
SQL> commit;
SQL> !
SQL> ! I should have a default (read only) transaction
SQL> !
SQL> show transaction
Transaction information:
Statement constraint evaluation is DEFERRED (off)
```

```
On the default alias
Transaction characteristics:
Read Write
```

```
Transaction information returned by base system:
no transaction is in progress
- transaction sequence number (TSN) 64 is reserved
- snapshot space for TSNs less than 64 can be reclaimed
- session ID number is 2
Session transaction modes (all)
Prestarted transactions are ENABLED for this session
Session user has a DEFAULT TRANSACTION
Snapshots are ENABLED IMMEDIATE
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.3.1. SQL no longer overrides the DEFAULT characteristic and each attach (connect) will use one of the following for the DECLARE DEFAULT TRANSACTION:

- If the user has a profile that includes a DEFAULT TRANSACTION clause, then those transaction characteristics will be used. This would require that the database administrator had executed a CREATE PROFILE and a CREATE USER statement to setup the default for the user.
- If there exists a DEFAULT PROFILE defined in the database, then those transaction characteristics will be used. This would require that the database administrator had executed the CREATE DEFAULT PROFILE statement to setup the defaults.
- Otherwise the default will be a READ ONLY transaction.

3.2.3 Unexpected Routine Call Failure Reporting RDB-E-OBSOLETE_METADATA

Bug 28255803

In Oracle Rdb Release V7.3.3, a change was made to correct an error calling SQL stored routines which executed transaction statements (SET TRANSACTION, START TRANSACTION, LOCK TABLE, COMMIT or ROLLBACK), used the GET DIAGNOSTICS statement (or the function SYS_GET_DIAGNOSTIC), or accessed declared local temporary tables and views.

These types of routines need the activating module name to be declared prior to executing the routine. This provides a required context for Oracle Rdb. The change involved a correction and retry if the reported missing module was that of the caller.

Unfortunately, the retry does not happen when the database has SYNONYMS enabled. The reported error that would trigger a retry is slightly different when a synonym for a module is not found and therefore the module was not being declared.

The following example shows the reported error.

```
%RDB-E-OBSOLETE_METADATA, request references metadata objects that no longer exist  
-RDMS-F-BAD_SYM, unknown module symbol - JAVA_RDB
```

This problem is known to affect Oracle JDBC for Rdb when attempting to connect to databases with synonyms enabled.

Databases without SYNONYMS enabled are not affected by this problem.

This problem has been corrected in Oracle Rdb Release 7.3.3.1.

3.2.4 Unexpected RDB-E-INVALID_BLR When Defining a Constraint

Bug 4461005

In prior releases of Oracle Rdb V7.3, it was possible that some constraint definitions would fail with an RDB-E-INVALID_BLR error. This internal error should never be reported and the cause is now fixed.

The following example shows a constraint defined on a version of the PERSONNEL database which fails in this way. Similar formulations of the constraint might show similar symptoms.

```

SQL> set dialect 'sql2011';
SQL>
SQL> create table EMPLOYEES (
cont>     EMPLOYEE_ID          CHAR (5)          PRIMARY KEY,
cont>     ...
cont>     BIRTHDAY              DATE,
cont>     STATUS_CODE           CHAR (1));
SQL>
SQL> create table JOB_HISTORY (
cont>     EMPLOYEE_ID          CHAR (5)          NOT NULL,
cont>     ...
cont>     DEPARTMENT_CODE       CHAR (4),
cont>     SUPERVISOR_ID          CHAR (5));
SQL>
SQL> alter table JOB_HISTORY
cont>     add constraint JH_EMP_FK2
cont>         check( (EMPLOYEE_ID is not null)
cont>                 or exists (select * from EMPLOYEES E
cont>                             where JOB_HISTORY.EMPLOYEE_ID = E.EMPLOYEE_ID)
cont>         )
cont>         initially immediate not deferrable
cont> ;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-INVALID_BLR, request BLR is incorrect at offset 38
SQL>

```

In this case, query rewrite eliminated the EXISTS clause and the reference to the EMPLOYEES table. The problem occurs when Oracle Rdb attempts to write dependency information for the constraint and notices that information for the EMPLOYEES table is not available.

A workaround to this problem is to execute SET FLAGS prior to the ALTER or CREATE TABLE Statement that adds the constraint which disables query rewrite.

```
SQL> set flags 'norewrite(is_null)';
```

This problem has been corrected in Oracle Rdb Release 7.3.3.1.

3.2.5 Unexpected Results for Queries Using Query Rewrite of AND/OR Operators

In prior releases of Oracle Rdb V7.3, it was possible for some queries to return wrong results when using query rewrite.

The following example shows a query which is transformed by the optimizer query rewrite facility and which incorrectly replaced the predicate with FALSE.

```

SQL> select count(*)
cont>   from EMPLOYEES E
cont>   where not
cont>     ( E.EMPLOYEE_ID is not null
cont>       and E.EMPLOYEE_ID in (select EMPLOYEE_ID from SUPERVISORS)
cont>     )
cont> ;
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (*) Q2

```

```

Conjunct: FALSE
Get      Retrieval sequentially of relation 0:EMPLOYEEES

          0
1 row selected
SQL>

```

This problem occurs when the NOT operator is applied to a predicate using AND or OR logical conditions and when one of the operand expressions is eliminated after applying nullability rules. In this example, EMPLOYEE_ID is part of a NOT DEFERRABLE PRIMARY KEY definition. From this, Rdb knows that the expression (E.EMPLOYEE_ID is not null) is always TRUE. Unfortunately, Rdb was applying the AND logic when, under the NOT operator, OR logic should have been applied.

If NOT NULL or PRIMARY KEY definitions are DEFERRABLE, then this problem is unlikely to occur as Rdb does not use DEFERRABLE constraints to gather nullability for a column.

A workaround to this problem is to execute SET FLAGS prior to executing the query (say before the OPEN of a cursor), or by defining RDMS\$SET_FLAGS prior to running the application.

```
$ DEFINE RDMS$SET_FLAGS "norewrite(is_null)"
```

This problem has been corrected in Oracle Rdb Release 7.3.3.1.

3.2.6 Unexpected Sequence Creation by DECLARE LOCAL TEMPORARY TABLE Statement

In prior releases of Oracle Rdb, the DECLARE LOCAL TEMPORARY TABLE statement incorrectly implemented IDENTITY columns. These columns may be defined explicitly using the IDENTITY clause or implicitly via a LIKE clause.

This is a list of unexpected problems that may be encountered.

- The DECLARE LOCAL TEMPORARY TABLE statement may fail during a READ ONLY transaction as shown in this example.

```

SQL> set transaction read only;
SQL>
SQL> declare local temporary table module.SAMPLE0
cont>      (sample_id bigint identity (cycle start with 2 increment by 2)
cont>      ,sample_description varchar (100)
cont>      )
cont>      comment is 'show that declare allows IDENTITY'
cont>      on commit delete rows
cont>      large memory is enabled
cont> ;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-READ_ONLY_TRANS, attempt to update during a read-only transaction
SQL>
SQL> rollback;
SQL>

```

This error is unexpected.

- Although the declared table is transitory and the name is disambiguated by the MODULE keyword, the DECLARE LOCAL TEMPORARY TABLE statement tries to define a permanent sequence to represent the IDENTITY column.

As shown in this example, this may fail due to an existing sequence.

```
SQL> set transaction read write;
SQL>
SQL> create sequence SAMPLE1;
SQL>
SQL> declare local temporary table module.SAMPLE1
cont>     (sample_id bigint identity (cycle start with 2 increment by 2)
cont>     ,sample_description varchar (100)
cont>     )
cont>     comment is 'show that declare allows IDENTITY'
cont>     on commit delete rows
cont>     large memory is enabled
cont> ;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-SEQEXTS, there is another sequence named "SAMPLE1" in this database
SQL>
SQL> rollback;
SQL>
```

- If there is no conflict during the DECLARE LOCAL TEMPORARY TABLE statement, then this leaves a system defined sequence which cannot be dropped. The following example shows an attempt to DROP this sequence.

```
SQL> drop sequence SAMPLE3 if exists;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-NOMETSYSREL, operation illegal on system defined metadata
-RDMS-E-SEQNOTDEL, sequence "SAMPLE3" has not been deleted
SQL>
```

Under normal circumstances, a DROP TABLE or an ALTER TABLE ... DROP COLUMN would also remove this sequence. However, these statements do not operate on declared temporary tables.

- Although no permanent table exists, attempts to create tables using this name will fail as shown in this example.

```
SQL> declare local temporary table module.SAMPLE4
cont>     (sample_id bigint identity (cycle start with 2 increment by 2)
cont>     ,sample_description varchar (100)
cont>     )
cont>     comment is 'show that declare allows IDENTITY'
cont>     on commit delete rows
cont>     large memory is enabled
cont> ;
SQL>
SQL> create table SAMPLE4
cont>     (sample_id bigint identity (cycle)
cont>     ,sample_description varchar (100)
cont>     )
cont> ;
%SQL-F-SEQALREXI, Sequence SAMPLE4 already exists in the schema or database
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.3.1. In this release of Oracle Rdb, the DECLARE LOCAL TEMPORARY TABLE statement will materialize a local (session) sequence matching the characteristics of the IDENTITY or GENERATED ... AS IDENTITY definition. Upon DISCONNECT, the

state of the sequence will be discarded and it will be reloaded for the next connection. Each temporary sequence is private to the connection.

3.2.7 Support for IDENTITY and Extended LIKE Table Clause for DECLARE LOCAL TEMPORARY TABLE

This release of Oracle Rdb enhances support in the DECLARE LOCAL TEMPORARY TABLE statement for column IDENTITY values. An IDENTITY or GENERATED ... AS IDENTITY clause creates a column that generates a unique integer for each inserted row.

This type of column is now supported for SQL for DECLARE LOCAL TEMPORARY TABLE. In this case, the system created sequence used for an identity column is also a LOCAL TEMPORARY and is materialized for each instantiation of the module.

The following example shows the explicit use of IDENTITY as well as the sequence attributes.

```
SQL> declare local temporary table module.SAMPLES_REC1
cont>      (sample_id generated always as identity (start with 100 cycle)
cont>      ,sample_desc varchar(30)
cont>      )
cont> ;
SQL>
SQL> insert into module.SAMPLES_REC1 values ('scratch sample #1');
1 row inserted
SQL> select * from module.SAMPLES_REC1
cont> ;
          SAMPLE_ID  SAMPLE_DESC
          100      scratch sample #1
1 row selected
SQL>
```

Additionally, DECLARE LOCAL TEMPORARY TABLE supports the enhanced LIKE table syntax. See [Section 6.1.12](#) for more details.

An IDENTITY column can also be inherited by a LIKE table clause, as shown in this example.

```
SQL> -- the IDENTITY here is *not* shared with the temporary table
SQL> create table SAMPLES_REC
cont>      (sample_id generated always as identity (start with 1 cycle)
cont>      ,sample_desc varchar(30)
cont>      );
SQL>
SQL> -- the IDENTITY defined here will become a scratch sequence
SQL> -- not shared with the base table
SQL> declare local temporary table module.SAMPLES_REC2
cont>      (like SAMPLES_REC including IDENTITY)
cont>      on commit delete rows
cont> ;
SQL>
SQL> insert into module.SAMPLES_REC2 values ('scratch sample #2');
1 row inserted
SQL> select * from module.SAMPLES_REC2
cont> ;
          SAMPLE_ID  SAMPLE_DESC
          1          scratch sample #2
1 row selected
```

SQL>

3.3 RMU Errors Fixed

3.3.1 Incorrect UNION Syntax Generated by RMU Extract for Some View Definitions

Bug 4461005

In prior releases of Oracle Rdb, the RMU Extract command did not correctly extract view definitions that contained UNION, EXCEPT, and INTERSECT operators. Attempts to use the SQL script reported syntax errors.

When the clauses ORDER BY, LIMIT TO, OFFSET, FETCH FIRST (or FETCH NEXT) appear within the select expression for a UNION ALL, UNION DISTINCT, EXCEPT DISTINCT (also known as MINUS) or INTERSECT DISTINCT operator, then the SQL syntax requires parentheses around that SELECT, otherwise these clauses signify the end of the operator.

The following example shows the problem with the missing parentheses.

```
$ rmu/extract/item=view/option=(match=v00,noheader,filename) 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 PERSONNEL';
create view V00
  (EMPLOYEE_ID,
   START_DATE,
   END_DATE,
   TABLE_ABBREV) as
select C2.EMPLOYEE_ID, C2.JOB_START, C2.JOB_END, 'JH'
  from JOB_HISTORY C2
  order by C2.EMPLOYEE_ID asc
union all
select C3.EMPLOYEE_ID, C3.SALARY_START, C3.SALARY_END, 'SH'
  from SALARY_HISTORY C3
  order by C3.EMPLOYEE_ID asc;

commit work;
$
```

This problem has been corrected in Oracle Rdb Release 7.3.3.1. RMU Extract now detects these clauses and correctly adds the required syntax as shown in this revised output.

```
$ rmu/extract/item=view/option=(match=v00,noheader,filename)-
/output=q.sql -
sql$database
$ sql$ @q
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 PERSONNEL';
SQL> create view V00
```

```

cont>      (EMPLOYEE_ID,
cont>        START_DATE,
cont>        END_DATE,
cont>        TABLE_ABBREV) as
cont>      (select C2.EMPLOYEE_ID, C2.JOB_START, C2.JOB_END, 'JH'
cont>        from JOB_HISTORY C2
cont>        order by C2.EMPLOYEE_ID asc)
cont>      union all
cont>      (select C3.EMPLOYEE_ID, C3.SALARY_START, C3.SALARY_END, 'SH'
cont>        from SALARY_HISTORY C3
cont>        order by C3.EMPLOYEE_ID asc);
SQL>
SQL> commit work;
$

```

3.3.2 RMU Show User Now Reports if the Database is Opened for RESTRICTED Access

In prior versions of Oracle Rdb, a database could be opened for RESTRICTED access (allowing only privileged users access) but the RMU Show User command did not indicate this state of the database.

The following example shows that RMU now displays this information.

```

$ rmu/open mf_personnel/acc=restricted
$ rmu/show user
Oracle Rdb V7.3-31 on node QWKBFX  7-NOV-2018 13:03:09.06
  - monitor started  7-NOV-2018 13:00:08.35 (uptime 0 00:03:00)
  - monitor log filename is "$1$DGA231:[RDMMON_LOGS]RDMMON731_QWKBFX.LOG;94"

database _$1$DGA174:[TESTING.V73.SHOW_USER]MF_PERSONNEL.RDB;1
  - opened  7-NOV-2018 13:02:56.01 (elapsed 0 00:00:13)
  * database is opened by an operator
  - database opened for RESTRICTED access
$

```

This problem has been corrected in Oracle Rdb Release 7.3.3.1.

3.4 RMU Show Statistics Errors Fixed

3.4.1 RMU SHOW STATISTICS LOGICAL AREA STATISTICS Zoom Menu %SMG-F-INVROW Error

Bug 28430946

When the Oracle Rdb RMU SHOW STATISTICS command is executed to display Rdb database statistics, the menu displayed when the Zoom screen option is selected for the "Logical Area Statistics" screen for a table logical area is supposed to overlay the statistic names on the left side of the screen, but the starting position of the menu was not adjusted when the Logical Area Statistics screen changed to accommodate terminal page sizes less than 48 rows.

For terminal page sizes less than 29 rows, this caused the following fatal error to be output when the Zoom option was selected on the bottom of the Logical Area Statistics screen.

```
%SMG-F-INVROW, invalid row
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-F-FTL_SHOW, Fatal error for SHOW operation at 13-AUG-2018 15:55:19.93
```

For terminal page sizes between 29 and 33 rows, the Zoom menu would be displayed but not all of the Zoom menu entries would appear on the screen. For all terminal page sizes between 29 and 47 rows, the menu position would be too low on the screen for the menu statistic choices to directly overlay the statistic names on the left side of the Logical Area Statistics screen.

These problems have been fixed. The starting position of the Zoom option menu for the Logical Area Statistics screen will now change to accommodate all valid terminal page sizes accepted by the Logical Area Statistics screen.

The following example shows the problems. The terminal page size is set to 28 rows. When the RMU/SHOW STATISTICS command displays the Logical Area Statistics screen for a database table logical area and the Zoom option is selected at the bottom of the screen, a fatal error is displayed and the RMU/SHOW STATISTICS session is terminated. When the terminal page size is then set to 29 rows and the Zoom option is selected, a menu of statistics names is displayed but it is not complete and the statistic menu choices do not overlay the same statistic names on the left side of the screen.

```
$set terminal/page=28
$rmu/show statistics mf_personnel
```

```
Node: TSTNOD (1/1/16)                               Oracle Rdb V7.3-300 Perf. Monitor
Rate: 3.00 Seconds                                  Logical Area Statistics
Page: 1 of 1                                         DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
.....
Table CANDIDATES in RDB$SYSTEM
```

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

record marked                0      0      0.0      0      0.0
```

Oracle® Rdb for OpenVMS

record fetched	0	0	0.0	0	0.0
fragmented	0	0	0.0	0	0.0
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	0	0	0.0	0	0.0
fragmented	0	0	0.0	0	0.0
sequential scan	0	0	0.0	0	0.0
record fetched	0	0	0.0	0	0.0

.....
 Config Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot Zoom !

%SMG-F-INVROW, invalid row
 %RMU-F-FATALOSI, Fatal error from the Operating System Interface.
 %RMU-F-FTL_SHOW, Fatal error for SHOW operation at 13-AUG-2018 15:55:19.93

\$set terminal/page=29
 \$rmu/show statistics mf_personnel

Node: TSTNOD (1/1/16)	Oracle Rdb V7.3-300 Perf. Monitor
Rate: 3.00 Seconds	Logical Area Statistics
Page: 1 of 1	DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1

.....
 Table CANDIDATES in RDB\$SYSTEM

statistic..... name.....	rate.per.second.....			total..... count.....	average..... per.trans....
	max.....	cur.....	avg.....		
record marked	0	0	0.0	0	0.0
record fetched	0	0	0.0	0	0.0
fragmented	0	0	0.0	0	0.0
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	0	0	0.0	0	0.0
fragmented	0	0	0.0	0	0.0
sequential scan	0	0	0.0	0	0.0
record fetched	0	0	0.0	0	0.0

.....
 Config Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot Zoom !

Node: TSTNOD (1/1/16)	Oracle Rdb V7.3-300 Perf. Monitor
Rate: 3.00 Seconds	Logical Area Statistics
Page: 1 of 1	DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1

Oracle® Rdb for OpenVMS

Table CANDIDATES in RDB\$SYSTEM

statistic..... name.....	rate.per.second.....			total..... count.....	average..... per.trans....
	max.....	cur.....	avg.....		
record marked	0	0	0.0	0	0.0
record fetched	0	0	0.0	0	0.0
fragmented	0	0	0.0	0	0.0
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	0	0	0.0	0	0.0
fragmented	0	0	0.0	0	0.0
A. record marked	0	0	0.0	0	0.0
B. record fetched	0	0	0.0	0	0.0
C. fragmented					
D. record stored					
E. fragmented					
F. pages checked					
G. saved IO					
H. discarded					

letter> to select logical area statistics, <control-Z> to cancel

The following example shows that these problems have been fixed. The terminal page size is set to 28 rows. When the RMU/SHOW STATISTICS command displays the Logical Area Statistics screen for a database table logical area and the Zoom option is selected, a fatal error does not occur and the complete menu is displayed overlaying the statistics displayed on the left side of the screen. When the terminal page size is then set to 29 rows and the Zoom option is selected, the complete menu is displayed and again overlays the statistics displayed on the left side of the screen.

```
$set terminal/page=28
$rmu/show statistics mf_personnel
```

```
Node: TSTNOD (1/1/16)                               Oracle Rdb V7.3-310 Perf. Monitor
Rate: 3.00 Seconds                                  Logical Area Statistics
Page: 1 of 1                                         DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
```

Table CANDIDATES in RDB\$SYSTEM

statistic..... name.....	rate.per.second.....			total..... count.....	average..... per.trans....
	max.....	cur.....	avg.....		
record marked	0	0	0.0	0	0.0
record fetched	0	0	0.0	0	0.0
fragmented	0	0	0.0	0	0.0
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	0	0	0.0	0	0.0
fragmented	0	0	0.0	0	0.0
sequential scan	0	0	0.0	0	0.0
record fetched	0	0	0.0	0	0.0

Oracle® Rdb for OpenVMS

.....
 Config Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot Zoom !

Node: TSTNOD (1/1/16) Oracle Rdb V7.3-310 Perf. Monitor
 Rate: 3.00 Seconds Logical Area Statistics
 Page: 1 of 1 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1

.....
 Table CANDIDATES in RDB\$SYSTEM

statistic..... name.....	rate.per.second.....			total..... count.....	average..... per.trans....
	max.....	cur.....	avg.....		
A. record marked	0	0	0.0	0	0.0
B. record fetched	0	0	0.0	0	0.0
C. fragmented	0	0	0.0	0	0.0
D. record stored	0	0	0.0	0	0.0
E. fragmented	0	0	0.0	0	0.0
F. pages checked	0	0	0.0	0	0.0
G. saved IO	0	0	0.0	0	0.0
H. discarded	0	0	0.0	0	0.0
I. record erased	0	0	0.0	0	0.0
J. fragmented	0	0	0.0	0	0.0
K. sequential scan	0	0	0.0	0	0.0
L. record fetched	0	0	0.0	0	0.0
M. ALL					

.....
 Type <return> or <letter> to select logical area statistics, <control-Z> to cancel

\$set terminal/page=29
 \$rmu/show statistics mf_personnel

Node: TSTNOD (1/1/16) Oracle Rdb V7.3-310 Perf. Monitor
 Rate: 3.00 Seconds Logical Area Statistics
 Page: 1 of 1 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1

.....
 Table CANDIDATES in RDB\$SYSTEM

statistic..... name.....	rate.per.second.....			total..... count.....	average..... per.trans....
	max.....	cur.....	avg.....		
record marked	0	0	0.0	0	0.0
record fetched	0	0	0.0	0	0.0
fragmented	0	0	0.0	0	0.0
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	0	0	0.0	0	0.0

Oracle® Rdb for OpenVMS

fragmented	0	0	0.0	0	0.0
sequential scan	0	0	0.0	0	0.0
record fetched	0	0	0.0	0	0.0

.....
 Config Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot Zoom !

Node: TSTNOD (1/1/16) Oracle Rdb V7.3-310 Perf. Monitor
 Rate: 3.00 Seconds Logical Area Statistics
 Page: 1 of 1 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1

.....
 Table CANDIDATES in RDB\$SYSTEM

statistic.....	rate.per.second.....			total.....	average.....
name.....	max.....	cur.....	avg.....	count.....	per.trans....
A. record marked	0	0	0.0	0	0.0
B. record fetched	0	0	0.0	0	0.0
C. fragmented	0	0	0.0	0	0.0
D. record stored	0	0	0.0	0	0.0
E. fragmented	0	0	0.0	0	0.0
F. pages checked	0	0	0.0	0	0.0
G. saved IO	0	0	0.0	0	0.0
H. discarded	0	0	0.0	0	0.0
I. record erased	0	0	0.0	0	0.0
J. fragmented	0	0	0.0	0	0.0
K. sequential scan	0	0	0.0	0	0.0
L. record fetched	0	0	0.0	0	0.0
M. ALL					

.....
 Type <return> or <letter> to select logical area statistics, <control-Z> to cancel

This problem has been corrected in Oracle Rdb Release 7.3.3.1.

3.5 Oracle Trace Errors Fixed

3.5.1 Oracle TRACE Data Not Being Collected for All Database Connections

Bug 9728436

In prior releases, Oracle Rdb failed to record Oracle TRACE (EPC) workloads for a second or subsequent database attach; via ATTACH, or CONNECT statements. Only workload performance statistics were collected for the initial database attach.

This problem has been corrected in Oracle Rdb Release 7.3.3.1. Oracle Rdb now correctly initializes the Oracle TRACE interface for subsequence attaches so that complete workloads are collected.

Chapter 4

Software Errors Fixed in Oracle Rdb Release 7.3.3.0

This chapter describes software errors that are fixed by Oracle Rdb Release 7.3.3.0.

4.1 Software Errors Fixed That Apply to All Interfaces

4.1.1 Unexpected Bugcheck After an Exception is Reported Inserting LIST OF BYTE VARYING Column

In prior releases of Oracle Rdb, it was possible that a runtime failure during the insert into a LIST OF BYTE VARYING column might generate a bugcheck. This occurs when there are many segments of the list buffered when an exception is reported (such as "storage area is full").

The following example shows the behavior using INSERT ... FILENAME statement. However, it is also possible with customer applications that insert multiple segments into a LIST OF BYTE VARYING column or when executing RMU Load on data that contain LIST OF BYTE VARYING columns.

```
SQL> declare NEW_WORK insert only cursor
cont>     for select * from DETAILS;
SQL>
SQL> declare NEW_TEXT insert only list cursor
cont>     for select COMMENTS where current of NEW_WORK;
SQL> declare NEW_PDF insert only list cursor
cont>     for select MANAGER_COMMENTS where current of NEW_WORK;
SQL>
SQL> open NEW_WORK;
SQL> insert into cursor NEW_WORK default values;
1 row inserted
SQL> open NEW_TEXT;
SQL> insert into cursor NEW_PDF
cont>     filename '2WS2610_PDF.PDF' as character varying;
%RDB-E-IMP_EXC, facility-specific limit exceeded
-RDMS-E-LISTAREAFULL, the areas specified in the list storage map for
DETAILS.MANAGER_COMMENTS are full
-RDMS-F-STAREAFUL, storage area USER2:[TESTING]SEG_STRING_08_AREA.RDA;1 is full
SQL> close NEW_PDF;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file
RDBVMS_USER2:[SMITHI]RDSBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=04, virtual
address=000000005C47AB88, PC=0000000000000000, PS=00000000
SQL> close NEW_WORK;
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0.

4.1.2 Unexpected RDB-E-BAD_DPB_CONTENT Error During ATTACH to a Database

Bug 26680957

In prior releases of Oracle Rdb V7.3, it was possible that certain third party applications might fail with an RDB-E-BAD_DPB_CONTENT error. This occurred because the application was passing an illegal value in the database parameter block and this error was now being caught by Rdb V7.3 and later versions.

%RDB-E-BAD_DPB_CONTENT, invalid database parameters in the database parameter block (DPB)

This problem has been corrected in Oracle Rdb Release 7.3.3.0. To allow these legacy applications to run, Oracle Rdb now relaxes the range test for this parameter.

4.1.3 Possible Wrong Results From Partitioned Indices

Bug 26546363

In prior versions of Oracle Rdb, it was possible to get incorrect results from a query when using constant complex value expressions for the selection criteria that matched index columns used for partitioning.

This problem occurred when:

- An index column was compared with a complex value expression. An example would be a call to a deterministic SQL function passing constant (literal) parameters.
Note that using a literal value such as numeric literal, date literal or a string is not a problem, only more complex expressions.
- The chosen index was partitioned with more than one partition (that is used a WITH LIMIT OF clause).
- The WITH LIMIT OF clause used a list of more than one value.
- The strategy attempted to use a range list tactic.

The constant value expressions are normally lifted out of the main query to reduce CPU overhead but this was not being used for the index lookup in the reported case.

This problem has been corrected in Oracle Rdb Release 7.3.3.0.

4.1.4 Unexpected Results From Partitioned Index Query

Bug 5371154

In older versions of Oracle Rdb, it was permitted to create a partitioned SORTED index with a USING clause that did not use the leading segments of the index.

The following example shows that the USING clause does not include the leading segment of the index.

```
define index HISTORY_DETAIL_NDX_1
  for HISTORY_DETAIL
  duplicates are NOT allowed
  store
    using YY within
      HISTORY_DETAIL_AREA_2
      with limit of '1996';
      HISTORY_DETAIL_AREA_1
  type is SORTED
  node size 430
  usage UPDATE.

  EMP_NO
  ascending.
  YY
```

```

    ascending.
MM
    ascending.
DED_CODE
    ascending.
end.

```

Such indices are not allowed in V7.0 and later versions because they prevent lookup optimizations for SORTED indices.

Unfortunately, if the database was created with an old version and converted (RMU Convert or RMU Restore) to Rdb V7.3, some queries could produce wrong results because sorted index optimizations were attempted and a partition scan was prematurely aborted.

This problem has been corrected in Oracle Rdb Release 7.3.3.0. Oracle Rdb now detects these types of index definitions and implicitly disables these sorted index optimizations to avoid this problem.

4.1.5 Possible Lost Database Updates When Using RMU Backup Incremental

Bug 26864598

In prior releases of Oracle Rdb, it was possible that RMU Backup Incremental would incorrectly skip updated pages when both ROW CACHE and INCREMENTAL BACKUP SCAN OPTIMIZATION were enabled on the database. Please be aware that the INCREMENTAL BACKUP SCAN OPTIMIZATION is the default setting for new databases.

Note

The RMU Dump Header=BACKUP command can be used to determine the current setting for a database.

```

$ rmu/dump/header=BACKUP/output=temp.txt sql$database
$ search temp.txt "Fast incremental backup"
    - Fast incremental backup is enabled
$

```

The incremental backup scan optimization allows RMU Backup Incremental to skip over SPAM page ranges which have not been changed since the last FULL database backup. This optimization can save database I/O by quickly locating actively updated page ranges. In rare cases, the RCS (Row Cache Server) might checkpoint rows to the database and mark them with a TSN (transaction sequence number) that was older (smaller value) than that recorded for the FULL database backup. This would indicate to RMU that they need not be backed up during an incremental backup.

For any customer using ROW CACHE, INCREMENTAL BACKUP SCAN OPTIMIZATION and RMU Backup Incremental, Oracle recommends that this release be installed and that a new RMU Backup Incremental be performed immediately to replace any that was previously performed.

A workaround to this problem is to avoid the use of the incremental backup scan optimization. This can be done by disabling INCREMENTAL BACKUP SCAN OPTIMIZATION as shown below.

```
SQL> alter database
cont> filename ...
cont> no incremental backup scan optimization;
SQL>
```

Or, by specifying that this optimization not be used on the RMU Backup Incremental command using the /NoScan_Optimization qualifier.

This problem has been corrected in Oracle Rdb Release 7.3.3.0.

4.1.6 Deterministic Function Not Treated As Deterministic

Bug 26388701

In prior releases of Oracle Rdb, a query with a function defined as DETERMINISTIC, such as TO_TIMESTAMP, may slow down significantly when dynamic optimization is used. The problem can be observed by setting the flag 'WATCH_CALL' which shows the repeated call to the function.

See the following example.

```
SQL> set flags 'strategy,detail(2),watch_call';
SQL>
SQL> ! The following query evaluates the function
SQL> ! "TO_TIMESTAMP" three times instead of once for a DETERMINISTIC function
SQL> select prod_id1, prod_id2, sale_date
cont> from T1
cont>     where prod_id1 = 2
cont>           and prod_id2 > 9
cont>           and sale_date >= TO_TIMESTAMP('27-Jun-2017')
cont> order by prod_id2;
Tables:
  0 = T1
Sort: 0.PROD_ID2(a)
Leaf#01 BgrOnly 0:T1 Card=12
  Bool: (0.PROD_ID1 = 2) AND (0.PROD_ID2 > 9) AND (0.SALE_DATE >= TO_TIMESTAMP (
    '27-Jun-2017', X'6D697373696E67', X'6D697373696E67'))
  BgrNdx1 T1_NDX [2:1] Fan=14
    Keys: (0.PROD_ID1 = 2) AND (0.SALE_DATE >= TO_TIMESTAMP ('27-Jun-2017',
      X'6D697373696E67', X'6D697373696E67'))
~Xa: routine "(unnamed)", user=RDB_EXECUTE
~Xa: routine "TO_TIMESTAMP", user=RDB_EXECUTE
~Xa: routine "TO_TIMESTAMP", user=RDB_EXECUTE
~Xa: routine "TO_TIMESTAMP", user=RDB_EXECUTE
      ID1          ID2    SALE_DATE
      2             10    27-JUN-2017 00:00:00.00
      2             10    28-JUN-2017 00:00:00.00
2 rows selected
SQL>
```

This example uses the TO_TIMESTAMP function provided by OCI Services for Rdb, but could occur with other customer defined functions.

To work around this problem, use an alternative syntax for the time stamp value using a literal such as TIMESTAMP'2017-06-27' or DATE VMS'27-Jun-2017'. Alternatively, use the SET FLAGS command 'MAX_STABILITY' to turn off the dynamic optimizer.

This problem has been corrected in Oracle Rdb Release 7.3.3.0.

4.1.7 Unexpected Bugcheck Generated When RDMS\$RUJ References Bad File Specification

Bug 27390568

In prior releases of Oracle Rdb, data definition (DDL) commands might bugcheck if the logical RDMS\$RUJ was incorrectly defined.

The following example shows the case where an unknown directory is used in the file specification.

```
$ define/job rdms$ruj sys$disk:[bad]
$ SQL$
SQL> attach 'filename sql$database';
SQL>
SQL> create view dl as select * from departments;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]RDSBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=04, virtual
address=0000000000000000, PC=FFFFFFFF80650230, PS=0000001B
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0. This problem no longer causes a bugcheck but correctly reports the cause of the problem.

```
$ SQL$
attach 'filename sql$database';
create view dl as select * from departments;
%RDB-F-SYS_REQUEST, error from system services request
-RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-FILACCERR, error creating run-unit journal file
USER2:[BAD]PERSONNEL$04C7MNU3G05IGH3PVVN17TF2U7.RUJ;
-RMS-E-DNF, directory not found
-SYSTEM-W-NOSUCHFILE, no such file
rollback;
SQL>
```

4.1.8 Wrong Result From Aggregate Functions Using DISTINCT and FILTER Clauses

Bug 25604746

In prior releases of Oracle Rdb, aggregates using both DISTINCT and FILTER clauses might return the wrong result.

For example, the following queries should return one row of salary_amount 51712.00 but instead erroneously returns NULL.

```
SQL> select sum(distinct salary_amount)
cont> filter (where salary_end is null)
cont> from salary_history
cont> where employee_id = '00164'
```

```

cont> ;
Tables:
  0 = SALARY_HISTORY
Aggregate: 0:SUM (DISTINCT 0.SALARY_AMOUNT) Q2
           Bool: MISSING (0.SALARY_END)
Leaf#01 BgrOnly 0:SALARY_HISTORY Card=729
         Bool: 0.EMPLOYEE_ID = '00164'
         BgrNdx1 SH_EMPLOYEE_ID [1:1] Fan=17
         Keys: 0.EMPLOYEE_ID = '00164'

                NULL

1 row selected
SQL>
SQL> select SUM(distinct salary_amount)
cont>   filter (where salary_start = '14-Jan-1983')
cont> from salary_history
cont> where employee_id = '00164'
cont> ;
Tables:
  0 = SALARY_HISTORY
Aggregate: 0:SUM (DISTINCT 0.SALARY_AMOUNT) Q2
           Bool: 0.SALARY_START = '14-JAN-1983'
Leaf#01 BgrOnly 0:SALARY_HISTORY Card=729
         Bool: 0.EMPLOYEE_ID = '00164'
         BgrNdx1 SH_EMPLOYEE_ID [1:1] Fan=17
         Keys: 0.EMPLOYEE_ID = '00164'

                NULL

1 row selected
SQL>

```

The only workaround is to move the FILTER clause predicate into the WHERE clause.

```

SQL> select sum(distinct salary_amount)
cont> from salary_history
cont> where employee_id = '00164' and
cont>         salary_end is null
cont> ;
Tables:
  0 = SALARY_HISTORY
Aggregate: 0:SUM (DISTINCT 0.SALARY_AMOUNT) Q2
Leaf#01 BgrOnly 0:SALARY_HISTORY Card=729
         Bool: (0.EMPLOYEE_ID = '00164') AND MISSING (0.SALARY_END)
         BgrNdx1 SH_EMPLOYEE_ID [1:1] Fan=17
         Keys: 0.EMPLOYEE_ID = '00164'

                51712.00

1 row selected
SQL>
SQL> select SUM(distinct salary_amount)
cont> from salary_history
cont> where employee_id = '00164' and
cont>         salary_start = '14-Jan-1983'
cont> ;
Tables:
  0 = SALARY_HISTORY
Aggregate: 0:SUM (DISTINCT 0.SALARY_AMOUNT) Q2
Leaf#01 BgrOnly 0:SALARY_HISTORY Card=729
         Bool: (0.EMPLOYEE_ID = '00164') AND (0.SALARY_START = '14-JAN-1983')
         BgrNdx1 SH_EMPLOYEE_ID [1:1] Fan=17
         Keys: 0.EMPLOYEE_ID = '00164'

```

51712.00
1 row selected
SQL>

The problem was that the DISTINCT was processed prior to the FILTER, and so the intermediate result set was incorrect.

These problems have been corrected in Oracle Rdb Release 7.3.3.0.

4.1.9 LSEDIT Support Not Installed When Only DECSET License Was Found

Bug 25652409

With this release of Oracle Rdb, the installation procedure no longer checks for a valid license for the Language Sensitive Editor (LSEDIT) or similar product such as DECSET before providing LSE support. The partial LSEDIT support will be provided as is with no implied support from Oracle as long as LSE is available on the target system.

This problem has been corrected in Oracle Rdb Release 7.3.3.0.

If there is no LSE installed on your system or if there is no valid license, then the Rdb installation will display text similar to the following:

```
*****  
  
SQL is not being installed with Language-Sensitive Editor (LSE)  
support because there were problems with LSE on your system.  
The problem may be that LSE has not been started correctly.  
If you want the Language-Sensitive Editor support you might try  
the following:  
  
1. Try starting up LSE - @SYS$STARTUP:LSE$STARTUP  
2. Re-install  
  
*****
```

Otherwise, a successful installation will display this output:

```
*****  
  
SQL has been provided with Language-Sensitive Editor(LSE)  
support using the VMS LSE language.  
  
*****
```

4.1.10 Sub-optimal Performance Observed for COUNT Aggregate

Bug 27137371

In prior releases of Oracle Rdb, applications built using the SQL Precompiler or SQL Module language prior to Rdb V7.3.1 might execute with sub-optimal performance when run against Oracle Rdb V7.3.2 or later.

This problem was due to the COUNT (columnname) and COUNT (DISTINCT columnname) strategy no longer using an existing index but degenerating to sequential retrieval. This problem was due to a re-formulation of the COUNT expression in Rdb V7.3, and a case where the older formulation was not correctly handled.

If those modules showing this behavior are recompiled under Rdb V7.3.2 or later, then performance is corrected.

This problem has been corrected in Oracle Rdb Release 7.3.3.0. Both the new and old formulations of COUNT are correctly optimized by this release without requiring a rebuild of the application.

4.1.11 Query Outline With MANDATORY Not Aborting Query When Compliance Not Possible

Bug 27137371

When creating a query outline, the database administrator can specify COMPLIANCE MANDATORY to ensure that the chosen strategy is obeyed. In such cases, the query start (UPDATE, DELETE, CALL, OPEN of a cursor, etc) will fail with an error:

```
%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist  
-RDMS-F-OUTLINE_FAILED, could not comply with mandatory query outline directives
```

In some cases, for nested subqueries, this checking of compliance was not correctly enforced. This may lead some query outlines to be partially applied and the non-compliance not reported to the application.

This problem has been corrected in Oracle Rdb Release 7.3.3.0. The Rdb optimizer now propagates the non-compliance from the nested subquery.

This change may have two affects:

1. It is possible that some queries will now fail with the RDMS-F-OUTLINE_FAILED error when in previous versions they did not.
2. Query outlines defined as COMPLIANCE OPTIONAL that were improperly applied will now report the following message when SET FLAGS 'STRATEGY' is active. There may also be noticeable changes in strategy due to the corrected behavior of the optimizer.

```
~S: Full compliance with the outline was not possible
```

4.2 SQL Errors Fixed

4.2.1 Unexpected Favoring of the First Storage Area When Using FILL RANDOMLY Clause

Bug 25600019

In prior versions of Oracle Rdb, it was possible that the FILL RANDOMLY clause of the LIST storage map would not distribute LIST OF BYTE VARYING columns uniformly.

In a customer reported example, the FILL RANDOMLY clause was applied to four storage areas. The generated random number is reduced to a value between 0 and 3 (using the modulus function) and the generated values skew heavily toward zero. Thus the first storage area is used most of the time.

The following example shows an analysis of the LIST OF BYTE VARYING distribution based on the selected logical area of the primary segment database key. It can be seen that the first storage area is heavily favored as a target.

```
SQL> select show_area(COMMENTS), count(*)
cont> from SAMPLES
cont> group by show_area(COMMENTS)
cont> ;
```

63	196358
64	903
65	1872
66	867

```
4 rows selected
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0. Oracle has replaced the random number generator previously used by Rdb with a more modern version that does not behave the same way.

4.2.2 VARIANCE Now Returns Zero for Single Row Groups for Oracle Dialects

In prior versions of Oracle Rdb, the VARIANCE aggregate function would return NULL when the number of rows in the group was just one. While this is the correct result for ANSI and ISO SQL Language Dialects, it differs from a result from an Oracle Database which would return 0.

With this release of Oracle Rdb, the result in such cases will be zero when SET DIALECT establishes ORACLE LEVEL1, ORACLE LEVEL2, or ORACLE LEVEL3 as the dialect. Note that applications using OCI Services for Rdb have the dialect implicitly set during connection to the service.

The following example shows the different result after setting the dialect.

```
SQL> select variance (salary_amount)
cont> from salary_history
cont> where employee_id = '00164'
cont> and salary_end is null;
```



```

                NULL
1 row selected
SQL> set dialect 'oracle level3';
SQL> select variance (salary_amount)
cont> from salary_history
cont> where employee_id = '00164'
cont> and salary_end is null;

    0.0000000000000000E+000
1 row selected
SQL>

```

This problem has been corrected in Oracle Rdb Release 7.3.3.0. If an application wants the previous behavior, then using VAR_SAMP aggregate function will generate the same result on Oracle Rdb and the Oracle Database.

4.2.3 SQL IMPORT DATABASE Does Not Restore THRESHOLDS for LIST Storage Map

Bug 25689972

In prior releases of Oracle Rdb, the SQL IMPORT DATABASE statement did not redefine the THRESHOLDS defined for the LIST storage map areas.

This example shows the original LIST storage map.

```

SQL> create storage map LISTS_MAP
cont>     store LISTS
cont>     in BLOB_AREA
cont>     (thresholds are (50, 78, 95))
cont>     for (RESUMES.RESUME)
cont>     in RDB$SYSTEM
cont> ;
SQL>
SQL> show storage map LISTS_MAP;
LISTS_MAP
For Lists
Store clause:      STORE LISTS
                  in BLOB_AREA
                  (thresholds are (50, 78, 95))
                  for (RESUMES.RESUME)
                  in RDB$SYSTEM

Partition information for lists map:
Partition: (0) SYS_P00058
Storage Area: BLOB_AREA
          Thresholds are (50, 78, 95)
Partition: (0) SYS_P00001
Storage Area: RDB$SYSTEM
SQL>

```

After the IMPORT, the original source shows the thresholds but the partition output shows that they are lost.

```

SQL> import database from MYDB.RBR filename MYDB.RDB ;
SQL>
SQL> show storage map LISTS_MAP;

```

```

LISTS_MAP
For Lists
Store clause:      STORE LISTS
                   in BLOB_AREA
                   (thresholds are (50, 78, 95))
                   for (RESUMES.RESUME)
                   in RDB$SYSTEM

```

Partition information for lists map:

```

Partition: (0) SYS_P00057
Storage Area: BLOB_AREA
Partition: (0) SYS_P00001
Storage Area: RDB$SYSTEM

```

SQL>

A workaround to this problem is to create an IMPORT DATABASE script that includes the redefinition of the storage map.

```

$ rmu/extract-
  /item=(import,storage)-
  /option=(noheader,match=RDB$SEGMENTED_STRINGS%,filename_only) -
  mydb -
  /out=tst.sql

```

Edit the generated script to remove the lines ";---end import" and "commit work;". This script can be used to override the exported LIST storage map during the import. A message similar to "Definition of STORAGE MAP LISTS_MAP overridden" will be displayed.

This problem has been corrected in Oracle Rdb Release 7.3.3.0.

4.2.4 Unexpected ACCVIO When Using INSERT ... RETURNING Statement

Bug 25697836

In prior versions of Oracle Rdb, the INSERT ... RETURNING statement could ACCVIO if only simple literal values were returned.

The following shows a simple example.

```

SQL> create table TEST_TABLE (f1 integer);
SQL> insert into TEST_TABLE values(1) returning 1;
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=0000000000000034, PC=FFFFFFFF8040CC20, PS=0000001B
SQL> insert into TEST_TABLE values(1) returning 1+1;

```

2

```

1 row inserted
SQL>

```

The workaround is to return a complex expression or columns in the RETURNING clause.

This problem has been corrected in Oracle Rdb Release 7.3.3.0.

4.2.5 Unexpected RDB-F-REQ_WRONG_DB Error Reported by SHOW TRANSACTION

In prior versions of Oracle Rdb, the SQL SHOW TRANSACTION statement might fail with an RDB-F-REQ_WRONG_DB error if more than one database were attached and the current user had a profile defined in these databases. That profile might be assigned to the current user or might be the DEFAULT PROFILE, which is used when no profile is assigned.

The following example shows the failure.

```
SQL> show transaction
Transaction information:
    Statement constraint evaluation is DEFERRED (off)

On alias DB1
Transaction characteristics:
    Default
    Wait 200 seconds before timeout
    Read only

Transaction information returned by base system:
a snapshot transaction is in progress
- all transaction sequence numbers (TSNs) less than 64 are visible
- TSN 64 is invisible
- all TSNs greater than or equal to 65 are invisible
- session ID number is 2
Session transaction modes (all)
Prestarted transactions are ENABLED for this session
Session user has a DEFAULT TRANSACTION
Snapshots are ENABLED IMMEDIATE

On alias DB2
Transaction characteristics:
    Default
%RDB-F-REQ_WRONG_DB, database named in specified request is not a database
named in specified transaction
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0.

4.2.6 Unexpected RDB-E-SEQNONEXT When Assigning DEFAULT to an IDENTITY Column

In prior releases of Oracle Rdb, DEFAULT could not be assigned to a column with the IDENTITY attribute, even when the AUTO_OVERRIDE flag was set. The AUTO_OVERRIDE flag should allow values to be assigned to AUTOMATIC and IDENTITY columns in special cases.

The following example shows that after the SET FLAGS statement, a literal can be inserted but not DEFAULT, which should be equivalent to not specifying the column in the INSERT statement.

```
SQL> insert into T (id) values (99);
%RDB-E-READ_ONLY_FIELD, attempt to update the read-only field ID
SQL>
SQL> insert into T (id) values (default);
```

```
%RDB-E-SEQNONEXT, The next value for the sequence "T" is not available
SQL>
SQL> set flags 'AUTO_OVERRIDE';
SQL>
SQL> insert into T (id) values (99);
1 row inserted
SQL>
SQL> insert into T (id) values (default);
%RDB-E-SEQNONEXT, The next value for the sequence "T" is not available
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0. Oracle Rdb now correctly applies the AUTO_OVERRIDE flag in this case; for both INSERT and UPDATE statements.

4.2.7 Unexpected Bugcheck When Executing Query With MIN, MAX or COUNT Functions

Bug 26109125

In Oracle Rdb Release 7.3.2.1, a problem was introduced that caused a bugcheck when multiple aggregates appeared in the same select list.

The following example shows this behavior.

```
SQL> select min(acct_nr), max(acct_nr) from acct_report;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTER]RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTER]RDSBUGCHK.DMP;
%RDB-F-BUG_CHECK, internal consistency check failed
SQL>
```

This occurred under the following conditions:

- The column was the leading field of an index,
- The table included a constraint on the column; either PRIMARY KEY or NOT NULL that was evaluated as NOT DEFERRABLE,
- The aggregate functions included MAX, MIN or COUNT and the optimizer tried to apply the multi-aggregate optimization.

A workaround for this problem is to define the flag NOCOUNT_SCAN, either using the SET FLAGS statement or defining the logical name RDMS\$SET_FLAGS.

This problem has been corrected Oracle Rdb Release 7.3.3.0.

4.2.8 SQL Precompiler Not Setting Exit Status When INCLUDE File Missing

Bug 18096215

In prior releases of Oracle Rdb, the SQL Precompiler did not correctly set the exit status when an INCLUDE statement was unable to locate the referenced file. This failure in turn meant that SQL started a host language compile with an incomplete source.

The following example shows that an incorrect exit status is returned. The status should indicate that a severe error was detected.

```
$ sql$pre/cc test.sc
%SQL-F-NO_INCFND, Could not find file missing_header.h named in INCLUDE
statement
-RMS-E-FNF, file not found
$ show symbol $severity
  $SEVERITY == "1"
$ show symbol $status
  $STATUS == "%X10000001"
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0. SQL now catches the exception and correctly sets the exit status for DCL (\$STATUS and \$SEVERITY) so that command procedures can detect the error and branch to some form of error handling. Additionally, the host language compile phase is skipped in such cases.

4.2.9 Unexpected Failure When WITH Clause Used With UNION Operator

Bug 24765398

In prior releases of Oracle Rdb V7.3, the WITH clause was applied only to the first SELECT clause within a UNION, MINUS, INTERSECT or EXCEPT operation.

The following example shows that the subquery factors (EMP_ID_LOW, EMP_ID_MID, and EMP_ID_OVER) are not visible to the other SELECT clauses.

```
SQL> with emp_id_low as
cont>   (select employee_id, last_name, first_name
cont>     from employees
cont>     where employee_id <= '02000')
cont> ,emp_id_mid as
cont>   (select employee_id, last_name, first_name
cont>     from employees
cont>     where employee_id > '02000' and employee_id <= '04000')
cont> ,emp_id_over as
cont>   (select employee_id, last_name, first_name
cont>     from employees
cont>     where employee_id > '04000')
cont> select * from emp_id_low
cont>   union
cont> select * from emp_id_mid
cont>   union
cont> select * from emp_id_over
cont>   ;
%SQL-I-VARNOTUSED, Variable "EMP_ID_OVER" was declared but never used
%SQL-I-VARNOTUSED, Variable "EMP_ID_MID" was declared but never used
%SQL-F-RELNOTDEF, Table EMP_ID_MID is not defined in database or schema
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0. SQL now correctly extends the scope to the entire operation.

4.2.10 Incorrectly Generated Routine Prototypes for Internal Calls in C++

In prior releases of Oracle Rdb, the generated prototypes for C++ might be malformed. This causes compiler errors when using the C++ compiler.

The following example shows that SQL Precompiler was incorrectly using the names of the input to the call as the name of the parameters.

```
void SQL$PRC2_1AKMV6VBKCC1GA5IK980 (
    int *SQLCODE                                /* out */
    ,void *dt.dt1                               /* out */
    .....^
    %CXX-E-EXPRPAREN, expected a ")"
    at line number 14 in file ...
    ,void *dt.dt2                                /* out */
);
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0. The parameters in the prototype declarations are now given generic names.

```
void SQL$PRC2_BURDQRVBKKCC1GA5IK980 (
    int *param1                                /* out */
    ,void *param2                              /* out */
    ,void *param3                              /* out */
);
```

4.2.11 Unexpected Value Inserted for Incorrect GUID Literal Value

Bug 27642467

In prior versions of Oracle Rdb, an incorrectly formatted GUID literal was not reported. Instead it was incorrectly replaced by the NIL GUID.

The following example shows that no error was reported because of an incorrect value. In this case, Oracle Rdb only supports UUID version 1 literals, and this literal is version 4.

```
SQL> insert into uuid_tab values (_guid'aa0058ed-bcca-4ead-a22f-b648ccb66787');
1 row inserted
SQL> select * from uuid_tab;
   A   B
   --  --
1  00000000-0000-0000-0000-000000000000
1 row selected
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0. Oracle Rdb now validates the passed literal and reports an error.

```
SQL> insert into uuid_tab values (_guid'aa0058ed-bcca-4ead-a22f-b648ccb66787');
%SQL-F-DATCONERR, Data conversion error for string
'aa0058ed-bcca-4ead-a22f-b648ccb66787'
```

```
-COSI-F-INVCVT, invalid data type conversion
SQL> select * from uuid_tab;
0 rows selected
SQL>
```

4.2.12 Unexpected RDB-E-OBSOLETE_METADA and RDMS-E-MODNEXTS Errors

Bugs 5726862 and 27802806

In some cases, when using GET DIAGNOSTICS, applications may fail with an RDB-E-OBSOLETE_METADATA error. The message would appear similar to the following example.

```
%RDB-E-OBSOLETE_METADATA, request references metadata objects that no longer exist
-RDMS-E-MODNEXTS, module RDB$DB_TEST does not exist in this database
```

The module named in the RDMS-E-MODNEXTS error is either a module name specified in the DECLARE MODULE statement, MODULE clause or a name created by prefixing the SQL precompiler source filename with RDB\$.

This error occurs when more than one module and more than one database are active in the application, and SQL neglects to declare the module context.

This problem has been corrected in Oracle Rdb Release 7.3.3.0. SQL now correctly declares a temporary context module for use by various SQL features, such as GET DIAGNOSTICS.

Although the error appears for SQL Precompiler or SQL Module Language applications, it occurs in the Dynamic SQL portion of the product and will be corrected without needing to rebuild the SQL application images.

4.2.13 Correction to Error Reporting of Database Not Yet Open

Bug 12735168

In prior versions of Oracle Rdb, attempts to attach to a database that was not OPEN caused Rdb to report this error:

```
%RDB-F-SYS_REQUEST, error from system services request
-RDMS-F-DBNOTOPEN, database is not open for access
```

When multiple databases are used by an application, it is hard to tell from this message which database has the issue. Additionally, older versions of Oracle Rdb reported this error as:

```
%RDB-E-NO_PRIV, privilege denied by database facility
-RDMS-F-DBNOTOPEN, database is not open for access
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0. Oracle Rdb now correctly reports a NO_PRIV error and includes the file specification used to open the database.

Oracle® Rdb for OpenVMS

```
$      RMU/OPEN RDM_OPEN_IS_MANUAL_7_DB1
$      SQL$
attach 'alias DB1 filename RDM_OPEN_IS_MANUAL_7_DB1';
attach 'alias DB2 filename RDM_OPEN_IS_MANUAL_7_DB2';
%SQL-F-ERRATTDEC, Error attaching to database RDM_OPEN_IS_MANUAL_7_DB2
-RDB-E-NO_PRIV, privilege denied by database facility
-RDMS-F-DBNOTOPEN, database is not open for access
-RDB-F-ON_DB, on database DISK:[DIRECTORY]RDM_OPEN_IS_MANUAL_7_DB2.RDB;1
```

4.2.14 Unexpected Data Type Result From BITSTRING Function

In prior versions of Oracle Rdb, the BITSTRING function was assigning the incorrect result type in some cases.

- The BITSTRING function was incorrectly assigning a date/time type when applied to a DATE, TIME, TIMESTAMP or INTERVAL data types.

```
SQL> select bitstring(current_timestamp from 1 for 64) from rdb$database;

      5-JUN-2018 11:01:24.90
1 row selected
SQL>
```

Since BITSTRING extracts a range of bits from the value, the source data type may not be applicable to the result. This has been corrected in this release. Oracle Rdb BITSTRING returns BIGINT.

- The BITSTRING function was incorrectly assigning a scale to the result when processing an integer type (TINYINT, SMALLINT, INTEGER, BIGINT) which was scaled. This is shown in the following example.

```
SQL> select bitstring(salary_amount from 1 for 32)
cont> from salary_history limit to 1 row;

      26291.00
1 row selected
```

Since BITSTRING extracts a range of bits from the value, the scale is meaningless for the result. This has been corrected in this release. Oracle Rdb BITSTRING returns only scale 0 results.

These problems have been corrected in Oracle Rdb Release V7.3.3.0.

4.3 RMU Errors Fixed

4.3.1 Handling RMU-F-AIJSEQAFT Error When Starting Continuous LogMiner

Bug 25881714

In previous versions of Oracle Rdb, when starting Continuous LogMiner with the command `RMU Unload After_Journal Continuous`, it might abort with the error `RMU-F-AIJSEQAFT`. This indicates that the listed after image journal backup files form an incomplete set.

```
%RMU-F-AIJSEQAFT, incorrect AIJ file sequence n when m was expected
```

This might have occurred because the list of journals, the options file or the wildcard specification did not select enough files when the `RMU Unload After_Journal` command began. In some cases, after the command started processing the backup journals, new backup files may have been created. After the known list of journals are processed, LogMiner enters a second phase which monitors changes in a Continuous mode. It is at the start of this second phase that the AIJ sequence numbers are validated.

With this release of Oracle Rdb, the `RMU Unload After_Journal` command has been enhanced to leave information for a possible restart when it aborts under this condition. RMU will define a DCL symbol `RMU$UNLOAD_RESTART` that can be used with the `/RESTART` qualifier on a subsequent `RMU` command.

DCL procedures which are executed to run Continuous LogMiner can be modified to retry if it fails with this error.

```
$ rmu/unload/after_journal ABC *.AIJ_BCK -
  /symbols -
  /order_aij_files -
  /table=(name=SAMPLE, output=SAMPLE.DAT) -
  /continuous
$ show symbol $status
$ if $status .eq. %X12C8AB24
$ then
$   ! %RMU-F-AIJSEQAFT, incorrect AIJ file sequence n when m was expected
$   ! Retry
$   show symbol rmu$unload*
$   rmu/unload/after_journal ABC *.AIJ_BCK -
     /symbols -
     /restart=&rmu$unload_restart -
     /order_aij_files -
     /table=(name=SAMPLE, output=SAMPLE.DAT) -
     /continuous
$ endif
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0.

4.3.2 Unexpected RDMS-E-BAD_CODE Reported When Using RMU Load to a LOCAL Temporary Table

In prior versions of Oracle Rdb, attempts to load into a LOCAL temporary table resulted in an unexpected *RDMS-E-BAD_CODE, corruption in the query string* error.

Both LOCAL or GLOBAL temporary tables should be valid staging tables for specialized loading of data. The intermediate staging table has an AFTER INSERT trigger that processes the data before it is inserted into one or more base tables. Temporary tables are obvious choices for staging tables as they can be defined as DELETE ROWS ON COMMIT, allowing release of memory based on the /COMMIT_EVERY qualifier of RMU Load.

The following example shows the reported error.

```
$ rmu/load/record=(file=sample,format=text) abc sample_loc sample.dat
  DEFINE FIELD IDENT DATATYPE IS TEXT SIZE IS 12.
  DEFINE RECORD SAMPLE.
    IDENT .
  END SAMPLE RECORD.
%RDMS-E-BAD_CODE, corruption in the query string
%RMU-I-DATRECREAD, 0 data records read from input file.
%RMU-I-DATRECSTO, 0 data records stored 5-JUL-2017 08:53:08.05.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 5-JUL-2017 08:53:08.05
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0. RMU now correctly establishes the environment for LOCAL temporary tables.

4.3.3 RMU/BACKUP/AFTER_IMAGE/FORMAT=NEW_TAPE AIJ Sequencing Problem

In prior versions of Oracle Rdb, the RMU Backup After_Image command would skip any empty fixed length journals. This only occurs when the /FORMAT=NEW_TAPE qualifier is used and results in an after image backup file which cannot be used to recover the database. This problem is most likely to occur when either /COMPRESSION or /ENCRYPT are used as these modes require the use of the /FORMAT=NEW qualifier.

When multiple fixed length After Image Journal (AIJ) files were backed up by the RMU Backup After_Image command to the same AIJ backup file using the /FORMAT=NEW_TAPE qualifier, and if any of the AIJ files contained valid Open records but did not contain any other AIJ records, they were ignored and not backed up. This created a problem later for RMU Recover because eliminating these AIJ files from the backup file could lead to fatal sequencing errors caused by the missing sequence numbers of the eliminated AIJ files when using the RMU Recover command.

This problem has been corrected in Oracle Rdb Release 7.3.3.0. RMU Backup After_Image now backs up all database fixed length AIJ files with valid Open records when the /FORMAT=NEW_TAPE qualifier is specified with the RMU Backup After_Image command.

Example Showing RMU Recover Problem

The following example shows the problem. The RMU/BACKUP/AFTER_IMAGE/FORMAT=NEW_TAPE command does not back up an AIJ file containing only a valid Open record with the sequence number "1" to

Oracle® Rdb for OpenVMS

the AIJ backup file but instead outputs the %RMU-I-EMPTYAIJ informational message. Later, when the AIJ backup file is recovered by the RMU/RECOVER/FORMAT=NEW_TAPE command, the recover operation is aborted because the AIJ Open record of the AIJ file that was not backed up by the RMU/BACKUP/AFTER_IMAGE/FORMAT=NEW_TAPE command contained the next sequence number "1" expected by the RMU/RECOVER/FORMAT=NEW_TAPE command.

```
$ rmu/backup/after_image/encrypt=name:hamlet/compress/log/format=new-
abc abc_save63_aij2.baij
%RMU-I-AIJBCKBEG, beginning after-image journal backup operation
%RMU-I-OPERNOTIFY, system operator notification: AIJ backup operation started
%RMU-I-AIJBCKSEQ, backing up after-image journal sequence number 1
%RMU-I-LOGBCKAIJ, backing up after-image journal AIJ2 at 14:38:34.28
%RMU-I-EMPTYAIJ, after-image journal file is empty
%RMU-I-AIJBCKSEQ, backing up after-image journal sequence number 2
%RMU-I-LOGBCKAIJ, backing up after-image journal AIJ1 at 14:38:34.33
%RMU-I-QUIETPT, waiting for database quiet point at 30-OCT-2017 14:38:34.33
%RMU-I-QUIETPTREL, released database quiet point at 30-OCT-2017 14:38:34.40
%RMU-I-LOGAIJBCK, backed up 3 committed transactions at 14:38:34.47
%RMU-I-OPERNOTIFY, system operator notification: AIJ backup operation completed
%RMU-I-AIJBCKEND, after-image journal backup operation completed successfully
%RMU-I-LOGAIJRN, backed up 2 after-image journals at 14:38:34.47
%RMU-I-LOGAIJBLK, backed up 148 after-image journal blocks at 14:38:34.47
%RMU-I-LOGAIJBCK, backed up 3 committed transactions at 14:38:34.47
%RMU-I-LOGCOMPR, data compressed by 81% (75427 Bytes in/14419 Bytes out)
%RMU-I-ENCRYPTUSED, Encryption key required when future restore performed.
$!
$! Later, when the database is recovered, a fatal sequence error is returned
$!
$ rmu/recover/encrypt=name:hamlet/noautomatic/log/format=new -
abc abc_save63_aij2.baij
%RMU-I-LOGRECDB, recovering database file DEVICE:[DIRECTORY]ABC.RDB;1
%RMU-F-AIJNORCVR, recovery must start with journal sequence 1
%RMU-F-FATALERR, fatal error on RECOVER
%RMU-F-FTL_RCV, Fatal error for RECOVER operation at 30-OCT-2017 14:38:39.32
```

Example showing the Corrected Behavior

The following example shows that this problem has been fixed. The RMU/BACKUP/AFTER_IMAGE/FORMAT=NEW_TAPE command now backs up an AIJ file containing only a valid Open record with the sequence number "1" to the AIJ backup file and also outputs the %RMU-I-EMPTYAIJ informational message. Later, when the AIJ backup file is recovered by the RMU/RECOVER/FORMAT=NEW_TAPE command, the recover operation succeeds because the AIJ Open record of the AIJ file that is now backed up by the RMU/BACKUP/AFTER_IMAGE/FORMAT=NEW_TAPE command contains the next sequence number "1" expected by the RMU/RECOVER/FORMAT=NEW_TAPE command.

```
$ rmu/backup/after_image/encrypt=name:hamlet/compress/log/format=new -
abc abc_save63_aij2.baij
%RMU-I-AIJBCKBEG, beginning after-image journal backup operation
%RMU-I-OPERNOTIFY, system operator notification: AIJ backup operation started
%RMU-I-AIJBCKSEQ, backing up after-image journal sequence number 1
%RMU-I-LOGBCKAIJ, backing up after-image journal AIJ2 at 14:40:29.63
%RMU-I-EMPTYAIJ, after-image journal file is empty
%RMU-I-AIJBCKSEQ, backing up after-image journal sequence number 2
%RMU-I-LOGBCKAIJ, backing up after-image journal AIJ1 at 14:40:29.67
%RMU-I-QUIETPT, waiting for database quiet point at 30-OCT-2017 14:40:29.67
%RMU-I-QUIETPTREL, released database quiet point at 30-OCT-2017 14:40:29.73
%RMU-I-LOGAIJBCK, backed up 3 committed transactions at 14:40:29.79
```

Oracle® Rdb for OpenVMS

```
%RMU-I-OPERNOTIFY, system operator notification: AIJ backup operation completed
%RMU-I-AIJBCKEND, after-image journal backup operation completed successfully
%RMU-I-LOGAIJRN, backed up 2 after-image journals at 14:40:29.80
%RMU-I-LOGAIJBLK, backed up 150 after-image journal blocks at 14:40:29.80
%RMU-I-LOGAIJBCK, backed up 3 committed transactions at 14:40:29.80
%RMU-I-LOGCOMPR, data compressed by 81% (76451 Bytes in/14621 Bytes out)
%RMU-I-ENCRYPTUSED, Encryption key required when future restore performed.
$!
$! Later, when the database is recovered, the recovery is successful
$!
$ rmu/recover/encrypt=name:hamlet/noautomatic/log/format=new -
  abc_save63_aij2.baij
%RMU-I-LOGRECDB, recovering database file DEVICE:[DIRECTORY]ABC.RDB;1
%RMU-I-LOGOPNAIJ, opened journal file DEVICE:[DIRECTORY]ABC_SAVE63_AIJ2.BAIJ
  at 30-OCT-2017 14:40:33.26
%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed
%RMU-I-LOGRECSTAT, transaction with TSN 162 ignored
%RMU-I-LOGRECSTAT, transaction with TSN 224 committed
%RMU-I-LOGRECSTAT, transaction with TSN 225 committed
%RMU-I-LOGRECSTAT, transaction with TSN 256 committed
%RMU-I-AIJONEDONE, AIJ file sequence 2 roll-forward operations completed
%RMU-I-LOGRECOVR, 3 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCE, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence number
  needed will be 3
%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 0 transactions ignored
%RMU-I-AIJSUCCE, database recovery completed successfully
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence number
  needed will be 3
```

4.3.4 Unexpected Bugcheck From RMU Verify Constraints

Bug 27817621

In prior versions of Oracle Rdb V7.3, it was possible that RMU Verify would generate a bugcheck when attempting to verify a constraint defined for a GLOBAL TEMPORARY table.

The following example shows the creation of the GLOBAL temporary table.

```
SQL> create global temporary table GBLTEMP_TBL (c1 integer);
SQL> alter table GBLTEMP_TBL
cont>      alter column c1
cont>          constraint GBLTEMP_TBL_NN
cont>              not null
cont>              initially immediate not deferrable;
SQL>
SQL> commit;
SQL>
```

This example shows the unexpected bugcheck when verifying the database.

Oracle® Rdb for OpenVMS

```
$ define rdms$set_flags "item_list"
$ rmu/verify/nolog/constraints personnel
~H Extension (VERIFY CONSTRAINTS) Item List: (len=0)
~H: ...verify constraint "COLLEGE_CODE_REQUIRED"
~H: ...verify constraint "DEPT_CODE_REQUIRED"
~H: ...verify constraint "EMPLOYEE_ID_REQUIRED"
~H: ...verify NOT NULL constraints for table "GBLTEMPBL"
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]RDSBUGCHK.DMP;
~H: ...verify constraint "JOB_CODE_REQUIRED"
~H: 5 tables, 7 constraints processed.
$
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0. The global temporary table is now skipped if it was not yet materialized.

A workaround to this problem is to fallback to the old verify constraint algorithms using the CONSTRAINT=FALLBACK keyword.

```
$ define rdms$set_flags "item_list"
$ rmu/verify/constraints=fallback/nolog personnel
~H Extension (VERIFY CONSTRAINTS) Item List: (len=0)
~H: ...verify constraint "COLLEGE_CODE_REQUIRED"
~H: ...verify constraint "DEPT_CODE_REQUIRED"
~H: ...verify constraint "EMPLOYEE_ID_REQUIRED"
~H: ...verify constraint "JH_EMP_ID_EXISTS"
~H: ...verify constraint "JOB_CODE_REQUIRED"
~H: ...verify constraint "SH_EMP_ID_EXISTS"
~H: ...verify constraint "GBLTEMPBL_NN"
$
```

4.3.5 Unexpected Area Corruption After RMU Move_Area of a UNIFORM Format Area

Bug 27670040

In prior versions of Oracle Rdb, the RMU Move_Area command with the qualifier BLOCKS_PER_PAGE might erroneously change the "clump page count" for a UNIFORM logical area rendering the logical area map in the SPAM page invalid. The following example shows the error received from such a command.

```
$ RMU/MOVE_AREA TST UNI_AREA_02/BLOCKS_PER_PAGE=16
%RMU-I-MOVTXT_04, Starting move of storage area
DISK:[DIRECTORY]UNI_AREA_02.RDA;2 at 16-MAR-2018 10:29:28.12
%RMU-W-BADPTLARE, invalid larea for uniform data page 4 in storage area 2
%RMU-W-BADPTLAR2, SPAM larea_dbid: 60, page larea_dbid: 61
%RMU-W-BADPTLARE, invalid larea for uniform data page 6 in storage area 2
%RMU-W-BADPTLAR2, SPAM larea_dbid: 61, page larea_dbid: 60
%RMU-W-BADPTLARE, invalid larea for uniform data page 7 in storage area 2
%RMU-W-BADPTLAR2, SPAM larea_dbid: 61, page larea_dbid: 60
%RMU-W-BADPTLARE, invalid larea for uniform data page 10 in storage area 2
%RMU-W-BADPTLAR2, SPAM larea_dbid: 60, page larea_dbid: 61
%RMU-W-BADPTLARE, invalid larea for uniform data page 16 in storage area 2
%RMU-W-BADPTLAR2, SPAM larea_dbid: 61, page larea_dbid: 0
%RMU-I-MOVTXT_01, Completed move of storage area
DISK:[DIRECTORY]UNI_AREA_02.RDA;2 at 16-MAR-2018 10:29:28.13
%RMU-I-MOVTXT_05, Moved snapshot area file
DISK:[DIRECTORY]UNI_AREA_02.SNP;2
```

Oracle® Rdb for OpenVMS

```
%RMU-I-LOGINIFIL,      contains 101 pages, each page is 16 blocks long
%RMU-I-MOVTEXT_15,    Live area files moved
%RMU-I-MOVTEXT_06,    Database root updated for all moved areas
%RMU-I-LOGDELFIL,     deleted file DISK:[DIRECTORY]UNI_AREA_02.RDA;1
%RMU-I-LOGDELFIL,     deleted file DISK:[DIRECTORY]UNI_AREA_02.SNP;1
%RMU-I-MOVTEXT_07,    Obsolete files deleted
%RMU-W-DOFULLBCK,     full database backup should be done to ensure future recovery
%RMU-I-COMPLETED,    MOVE_AREA operation completed at 16-MAR-2018 10:29:28.17
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0. RMU Move_Area now correctly handles the BLOCKS_PER_PAGE qualifier for UNIFORM page format areas.

4.3.6 RMU Extract Generates an Extraneous Semicolon on DISABLE PRIMARY KEY Clause

Bug 27927704

In prior releases of Oracle Rdb, RMU Extract would emit an extraneous semicolon. The following example shows the error reported when executing the generated script.

```
@BEFORE.SQL
SQL> set verify;
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 MYDB';
SQL> create table MYTAB (
cont>     COL1
cont>         INTEGER
cont>         default 0
cont>         constraint MYTAB_CONSTRAINT_NOT_NULL1
cont>             not null
cont>             initially deferred deferrable,
cont>     COL2
cont>         DATE VMS
cont>         default CURRENT_TIMESTAMP
cont>         constraint MYTAB_CONSTRAINT_NOT_NULL2
cont>             not null
cont>             initially deferred deferrable,
cont>     constraint MYTAB_PRIMARY1
cont>         primary key (COL1, COL2)
cont>         initially deferred deferrable)
cont>     disable constraint MYTAB_CONSTRAINT_NOT_NULL1
cont>     disable constraint MYTAB_CONSTRAINT_NOT_NULL2
cont>     disable primary key;;
        disable primary key;;
                ^
%SQL-F-JUNONLIN, Extraneous characters found after the end of the statement
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0.

4.3.7 RMU Dump Audit Always Assumed Type=ALL Even When Audit Classes Were Specified

Oracle® Rdb for OpenVMS

In prior releases of Oracle Rdb 7.3, the /TYPE qualifier of the RMU Dump Audit command incorrectly defaulted to ALL even when a subset of audit classes were specified.

This problem has been corrected in Oracle Rdb Release 7.3.3.0. If you omit the /TYPE qualifier, the default is ALL. If you specify a list of audit classes, for instance, /TYPE=(PROTECT,DACCESS) then only those classes are now dumped.

4.4 LogMiner Errors Fixed

4.4.1 LogMiner Extracts Incorrect TSN Values From After Image Journal

Bug 27371005

In prior releases of Oracle Rdb, the LogMiner interface (RMU Unload After_Image) incorrectly extracts TSN values from the after image journal. Specifically, as TSN values grow to occupy 32 bits or more, the values may be extracted as negative values. It is also possible that some values larger than 32 bits will be truncated.

The following output (using /FORMAT=DUMP) demonstrates the negative TSN value (for the field RDB\$LM_TSN).

```
$ rmu/unload/after-
  /format=dump-
  /table=(name:COLLEGES,out:sys$output) -
  mf_personnel -
  aijbck.aij
%RMU-I-UNLAIJFL, Unloading table COLLEGES to SYS$OUTPUT.:
%RMU-I-LOGOPNAIJ, opened journal file
  USERS2:[TESTER]AIJBCK.AIJ;1 at 17-JAN-2018 15:43:25.58
%RMU-I-AIJRSTSEQ, journal sequence number is "2"
RDB$LM_ACTION           : M (Modify)
RDB$LM_RELATION_NAME    : COLLEGES
RDB$LM_RECORD_TYPE      : 37
RDB$LM_DATA_LEN         : 56
RDB$LM_NBV_LEN          : 5
RDB$LM_DBK              : 92:6:11
RDB$LM_START_TAD        : 17-JAN-2018 15:39:33.0789733
RDB$LM_COMMIT_TAD       : 17-JAN-2018 15:39:33.1149733
RDB$LM_TSN              : -2147483616
RDB$LM_REC_VER          : 1
COLLEGE_CODE            : COL1
COLLEGE_NAME            : College 1
CITY                     :
STATE                    :
POSTAL_CODE              :
.
.
.
```

The impact will depend upon the usage of these TSN values. There is no workaround for this problem. Oracle recommends that an upgrade to this release be completed as soon as possible.

This problem has been corrected in Oracle Rdb Release 7.3.3.0.

4.5 RMU Show Statistics Errors Fixed

4.5.1 RMU SHOW STATISTICS Screen Did Not Display No Cluster Support Warning

Bug 27378673

The RMU Show Statistics "Logical Area Overview" screen is one of the RMU Show Statistics screens that does not support cluster wide statistics or statistics from multiple nodes. These screens should all display the warning message

*** cluster statistics collection not available for this screen ***

on the bottom line of the screen for a limited time when invoked and when the screen display is refreshed, but that message was not displayed in this case.

This problem has been corrected in Oracle Rdb Release 7.3.3.0. The warning message will now be displayed.

4.6 Rdb SGA API Errors Fixed

4.6.1 RMUST_CT_LAPMS_STATS Class Returned Incorrect RMUST_T_LOGICAL_AREA_ID Value

The Oracle Rdb SGA API interface automates retrieving database statistics available only through the RMU Show Statistics command by providing a way to retrieve statistics for Oracle Rdb databases from an application. There was a problem where the Rdb SGA API RMUST_CT_LAPMS_STATS Class returned the wrong RMUST_T_LOGICAL_AREA_ID tag logical area id value. The value returned for this tag was one more than it should have been due to an initialization problem. For example, if the DEPARTMENTS table logical area id in the MF_PERSONNEL datababase was "74", the value returned in the result buffer following this tag was "75". The Oracle Rdb RMU/SHOW AIP command can be used to show the correct logical area ids for an Rdb database. This problem happened only if the RMUST_T_LOGICAL_AREA_ID tag did not specify the logical area id value for the logical area in the SGA interface request buffer so the SGA interface set the logical area id in the SGA result buffer, and only for the RMUST_CT_LAPMS_STATS Class, not for the other SGA logical area classes available for the SGA interface.

This problem has been fixed. Now if the RMUST_CT_LAPMS_STATS class is specified in the SGA request buffer with a RMUST_T_LOGICAL_AREA_ID tag that does not specify a logical area id value, the SGA interface will insert the correct logical area id following the RMUST_T_LOGICAL_AREA_ID tag in the returned result buffer.

The following example shows the problem. The request buffer for the RMUST_CT_LAPMS_STATS class specified the tags RMUST_T_LOGICAL_AREA_ID and RMUST_T_LAPMS. The LAPMS data is returned for the DEPARTMENTS table in logical area "74" in the result buffer following the RMUST_T_LAPMS tag, but the incorrect logical area id of "75" is returned following the RMUST_T_LOGICAL_AREA_ID tag in the result buffer.

```
RMUST_T_LOGICAL_AREA_ID: 75
RMUST_T_LAPMS: LAPMS data of length 320
```

The following example shows that this problem has been fixed. The request buffer for the RMUST_CT_LAPMS_STATS class specified the tags RMUST_T_LOGICAL_AREA_ID and RMUST_T_LAPMS. The LAPMS data is returned for the DEPARTMENTS table in logical area "74" in the result buffer following the RMUST_T_LAPMS tag, and the correct logical area id of "74" is returned following the RMUST_T_LOGICAL_AREA_ID tag in the result buffer.

```
RMUST_T_LOGICAL_AREA_ID: 74
RMUST_T_LAPMS: LAPMS data of length 320
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0.

4.6.2 RMUST_CT_LAPMS_STATS Class Returned Statistics for Unused Logical Areas

There was a problem where the Oracle Rdb SGA API RMUST_CT_LAPMS_STATS class, which should return the LAPMS statistics for all logical areas currently assigned to an Oracle Rdb database object (such as

an index or table) if a specific logical area id is not included in the request buffer, also retrieved LAPMS statistics for unused logical areas which were not currently assigned to a database object.

This problem has been fixed. Now if the RMUST_CT_LAPMS_STATS class is specified in the SGA request buffer with a RMUST_T_LOGICAL_AREA_ID tag that does not specify a logical area id value, the SGA interface will return only the LAPMS statistics for logical areas that are currently assigned to a database object. If the user specifies a logical area id value that requests a specific logical area, the statistics for that specific logical area will continue to be returned whether or not the logical area is currently in use. The RMU/SHOW AIP command can be used to list all the logical areas that are currently assigned to a database object.

The following example shows the problem. The request buffer for the RMUST_CT_LAPMS_STATS class specifies the tags RMUST_T_LOGICAL_AREA_ID and RMUST_T_LAPMS. The LAPMS statistics entries returned in the result buffer include not only statistics for the active logical areas for the RESUMES table with a logical area id of "94" and the SALARY_HISTORY table with a logical area id of "96", but also include statistics for the unused logical area with a logical area id of "95".

```
RMUST_T_LOGICAL_AREA_ID: 94
LAPMS data of length 320
-----
RMUST_T_LOGICAL_AREA_ID: 95
LAPMS data of length 320
-----
RMUST_T_LOGICAL_AREA_ID: 96
LAPMS data of length 320
```

The following example shows that this problem has been fixed. The request buffer for the RMUST_CT_LAPMS_STATS class specifies the tags RMUST_T_LOGICAL_AREA_ID and RMUST_T_LAPMS. The LAPMS statistics entries returned in the result buffer include only statistics for the active logical areas for the RESUMES table with a logical area id of "94" and the SALARY_HISTORY table with a logical area id of "96", not statistics for the unused logical area with a logical area id of "95".

```
RMUST_T_LOGICAL_AREA_ID: 94
LAPMS data of length 320
-----
RMUST_T_LOGICAL_AREA_ID: 96
LAPMS data of length 320
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0.

4.7 Oracle Trace Errors Fixed

4.7.1 EPC\$REGISTRAR Startup Access Violation if Unable to Access Rdb RUJ File

Bug 27458974

There was a problem where the Oracle Trace EPC\$REGISTRAR process failed with a SYSTEM-F-ACCVIO and an EPC\$BUGCHECK.DMP file was created because an Oracle Rdb RUJ file could not be accessed for transactions involving the Oracle Trace EPC\$ADMIN_DB or EPC\$HISTORY_DB databases. This could happen due to disk quotas being exceeded or if there was an invalid Oracle Rdb RUJ file specification. The Oracle Trace EPC\$HOME_DIR:EPC\$REGISTRAR_nodename.LOG showed the access violation and indicated that an EPC\$BUGCHECK.DMP file was created but did not output an error message indicating the cause of the problem. The EPC\$BUGCHECK.DMP file also did not indicate the actual cause of the problem.

This problem has been fixed. Now, instead of a general SYSTEM-F-ACCVIO error and an unnecessary EPC\$BUGCHECK.DMP file being created, a specific error message will be output which indicates the cause of the problem.

The following example shows the problem happening when the Oracle Trace Registrar process was started and an Oracle Rdb RUJ file could not be accessed for a transaction involving the Oracle Trace EPC\$ADMIN_DB database.

```
$ RUN SYS$SYSTEM:EPC$REGISTRAR.EXE
$! The EPC$HOME_DIR:EPC$REGISTRAR_nodename.LOG showed:
%EPC-E-BUGCHK, Fatal error encountered, a dump file is being generated
%SYSTEM-F-ACCVIO, access violation, reason mask=01, virtual
address=0000000000000000, PC=0000000000000000, PS=0000001B

Improperly handled condition, image exit forced by last chance handler.
Signal arguments:   Number = 0000000000000005
                   Name   = 000000000000000C
                   0000000000000001
                   0000000000000000
                   0000000000000000
                   0000000000000000
                   000000000000001B

Register dump:
R0 = 0000000000000000  R1 = 0000000000000000  R2 = 0000000000000000
R3 = 0000000100000025  R4 = 000000007FFCF818  R5 = 000000007FFCF8B0
R6 = 0000000010000001  R7 = 0000000000000001  R8 = FFFFFFFFFFFFFFFF8
R9 = 000000007AA34AA0  R10 = 0000000060420081  R11 = 000000007AA34A9A
SP = 000000007AA32050  TP = 000000007B5181C8  R14 = 0000000000000000
R15 = 000000007FFD06F0  R16 = 0000000000000000  R17 = 000000007B8EEA90
R18 = 0000000060420089  R19 = 0000000000000001  R20 = 000000000240088
R21 = 0000000000000000  R22 = FFFFFFFFFFFF47D3  R23 = 0000000000000000
R24 = 000000007B912100  R25 = 0000000000000002  R26 = 000000007B912100
R27 = 000000007AA31FE8  R28 = 000007FDBFF962C8  R29 = 0000000000005718
R30 = 000000007AA31E88  R31 = 0000000000050970  PC = 0000000000000000
BSP/STORE = 000007FDBFFD4D00 / 000007FDBFFD4BB8  PSR = 000101308426030
IIPA = FFFFFFFF842621F0
B0 = FFFFFFFF8461A830  B6 = FFFFFFFF842621A0  B7 = 0000000000000000
```

Oracle® Rdb for OpenVMS

Interrupted Frame RSE Backing Store, Size = 2 registers

R32 = 000000007AA32998 R33 = 000000007AA32140

\$! The EPC\$BUGCHECK.DMP file showed:

Stack Dump Summary

```
***** Exception at 00000000001B3480 : EPC$REGISTRAR\SQL$SIGNAL_STOP
Module SQL$GETERR + 00001D60; line 6780
```

The following example shows that this problem has been fixed. In this case, the EPC\$REGISTRAR process could not access an Rdb RUJ file for a transaction to the Oracle Trace EPC\$ADMIN_DB database because of an invalid RUJ file specification. The %SYSTEM-F-ACCVIO error does not occur and an unnecessary EPC\$BUGCHECK.DMP file is not created. Instead, an error message is output indicating an invalid RUJ file specification. This could also happen if the RUJ file could not be accessed because of disk quotas being exceeded.

```
$ RUN SYS$SYSTEM:EPC$REGISTRAR.EXE
%RDB-F-SYS_REQUEST, error from system services request
-RDMS-F-FILACCERR, error parsing file DISK:[DIRECTORY]
EPC$ADMIN_DB$04C7MNU3G05U2H3Q0G07M5QCNT.RUJ;
-COSI-I-NOTDISKFILE, file is not a disk file
```

This problem has been corrected in Oracle Trace Release 7.3.0.0.

Chapter 5

Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.2

5.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.2

5.1.1 New PCSI Support for Rdb Kit Installation and Deinstallation

Starting with Oracle Rdb Release 7.3.3.2, whenever Oracle Rdb is installed or deinstalled, Oracle Rdb will be registered in the PCSI software product database. This will allow users to use the PCSI PRODUCT SHOW HISTORY and PRODUCT SHOW PRODUCT commands to display information about releases of Oracle Rdb that have been installed or deinstalled. This information will also be helpful as input whenever a Service Request (SR) is submitted to Oracle Support.

The following lines will now be displayed during the installation of Oracle Rdb, showing that the installation has been registered in the PCSI database.

```
The following product has been selected:
  ORCL I64VMS RDB73 V7.3-320           Transition (registration)

The following product will be registered:
  ORCL I64VMS RDB73 V7.3-320           DISK$NODE84_2:[VMS$COMMON.]

File lookup pass starting ...

Portion done: 0%
...100%

File lookup pass completed search for all files listed in the product's PDF
Total files searched: 0  Files present: 0  Files absent: 0

The following product has been registered:
  ORCL I64VMS RDB73 V7.3-320           Transition (registration)
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target directories...
```

Registration in the PCSI software product database allows a user to use commands such as the following to track what Oracle Rdb releases are currently installed and the history of any past product installations and deinstallations.

```
$ PRODUCT SHOW HISTORY/SINCE
-----
PRODUCT                                KIT TYPE      OPERATION     VAL DATE
-----
ORCL I64VMS RDB73 V7.3-320             Transition    Reg Product  (U) 10-OCT-2019
-----

1 item found

$ PRODUCT SHOW HISTORY RDB7*
-----
PRODUCT                                KIT TYPE      OPERATION     VAL DATE
-----
ORCL I64VMS RDB73 V7.3-320             Transition    Reg Product  (U) 10-OCT-2019
-----
```

Oracle® Rdb for OpenVMS

1 item found

```
$ PRODUCT SHOW PRODUCT RDB7*
```

PRODUCT	KIT TYPE	STATE
ORCL I64VMS RDB74 V7.3-320	Transition	Installed

1 item found

The following lines will now be displayed during the deinstallation of Oracle Rdb, showing that the removal of the release has been registered in the PCSI database. Deinstallation is performed by executing the DCL procedure `SY$MANAGER:RDB$DEINSTALL_DELETE.COM`. Please refer to section "Deleting Versions of Oracle Rdb" in the Oracle Rdb Installation Guide for further details.

The following product has been selected:

```
ORCL I64VMS RDB73 V7.3-320          Transition (registration)
```

The following product will be removed from destination:

```
ORCL I64VMS RDB73 V7.3-320          DISK$CLYPPR84_2:[VMS$COMMON.]
```

Portion done: 0%...100%

The following product has been removed:

```
ORCL I64VMS RDB74 V7.3-320          Transition (registration)
```

The example below shows the additional information that will be displayed by the PCSI `PRODUCT` commands as a result of the deinstallation of a release of Oracle Rdb.

```
$ PRODUCT SHOW HISTORY/SINCE
```

PRODUCT	KIT TYPE	OPERATION	VAL	DATE
ORCL I64VMS RDB73 V7.3-320	Transition	Remove	-	10-OCT-2019
ORCL I64VMS RDB73 V7.3-320	Transition	Reg Product	(U)	10-OCT-2019

2 items found

```
$ PRODUCT SHOW HISTORY RDB7*
```

PRODUCT	KIT TYPE	OPERATION	VAL	DATE
ORCL I64VMS RDB73 V7.3-320	Transition	Remove	-	10-OCT-2019
ORCL I64VMS RDB73 V7.3-320	Transition	Reg Product	(U)	10-OCT-2019

2 items found

```
$ PRODUCT SHOW PRODUCT RDB7*
```

PRODUCT	KIT TYPE	STATE
---------	----------	-------

0 items found

5.1.2 Updated Support for RDMS\$BIND_CODE_OPTIMIZATION

This logical name can be used to enable an optimization on Oracle Rdb systems executing on emulated Alpha hardware. Please refer to the Oracle Rdb Release 7.3.3.1 release notes for further details.

This release includes the following improvements: All generated code is now stored in separate code pages. This positively effects the Alpha hardware emulators by ensuring that the generated code is read only. That is, it avoids re-emulation because of updates to adjacent data portions of the page.

5.1.3 DUPLICATES ARE NOT ALLOWED Clause Added to ALTER INDEX Statement

This release of Oracle Rdb adds the clause DUPLICATES ARE NOT ALLOWED to the ALTER INDEX Statement. This is the inverse of the existing DUPLICATES ARE ALLOWED clause.

This new clause first verifies that the index (SORTED, SORTED RANKED, or HASHED), in fact, has no duplicate values.

Note

This command will perform I/O to the index but in general will be less costly in terms of I/O and CPU usage compared to equivalent DROP INDEX and CREATE UNIQUE INDEX statements. Oracle recommends using SET FLAGS 'INDEX_STATS' prior to performing the ALTER INDEX Statement to gather useful information on the progress of the statement.

If the scan detects any duplicate values, the scan is terminated and the actions of the ALTER INDEX Statement (if any) are rolled back.

The following example shows the reported error message when the ALTER INDEX statement cannot change the index to be unique.

```
SQL> create index mi_ndx on employees (middle_initial);
SQL>
SQL> --> should fail because middle_initial has duplicates
SQL> alter index mi_ndx
cont>     duplicates are NOT allowed
cont> ;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-DUPNOTALL, duplicate records not allowed for index MI_NDX
-RDMS-E-IDXNOTCHG, index MI_NDX has not been changed
SQL>
```

If no duplicate values are found, then the index is converted to a UNIQUE index.

5.1.4 New RMU/SET_DATABASE Qualifier /ACCESS=[UN]RESTRICTED

The new Oracle Rdb RMU SET DATABASE command ACCESS=[UN]RESTRICTED qualifier sets or clears the restricted database access flag in the database root file if the database is currently open or closed. If the restricted database access flag in the database root file is set, SQL users must have DBADM privilege to attach to an Rdb database. Previously, the restricted database access flag could only be cleared or set by the

RMU OPEN command ACCESS=[UN]RESTRICTED qualifier when it opened an Rdb database. If the restricted access bit is set in the database root, the RMU DUMP/HEADER command will show the following output when it displays the root parameters of an Rdb database. The ACCESS=[UN]RESTRICTED qualifier requires that the user has been granted RMU\$ALTER privilege.

```
"Access restricted to privileged users"
```

The command line syntax for the RMU SET DATABASE command ACCESS qualifier is:

```
/ACCESS=[UN]RESTRICTED
```

If ACCESS=RESTRICTED is specified, the restricted database access flag in the database root file is set. If ACCESS=UNRESTRICTED is specified, the restricted database access flag in the database root file is cleared. RESTRICTED or UNRESTRICTED must be specified.

In the following example, when /ACCESS=RESTRICTED is specified in the RMU/SET DATABASE command, the RMU/DUMP/HEADER command displays "Access restricted to privileged users" to confirm that the restricted database access flag in the database root has been set. When ACCESS=UNRESTRICTED is specified, the RMU/DUMP/HEADER command display of the root parameters does not include "Access restricted to privileged users", confirming that the restricted database access flag in the database root has been cleared.

```
$ RMU/SET DATABASE/ACCESS=RESTRICTED MF_PERSONNEL
%RMU-I-MODIFIED, Database state modified
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
$ RMU/DUMP/HEADER/OUT=MFP.HDR MF_PERSONNEL
$ SEAR MFP.HDR "ACCESS RESTRICTED"
    Access restricted to privileged users
$!
$ RMU/SET DATABASE/ACCESS=UNRESTRICTED MF_PERSONNEL
%RMU-I-MODIFIED, Database state modified
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
$ RMU/DUMP/HEADER/OUT=MFP.HDR MF_PERSONNEL
$ SEAR MFP.HDR "ACCESS RESTRICTED"
%SEARCH-I-NOMATCHES, no strings matched
```

5.1.5 New RMU/VERIFY Qualifier /VALIDATE for FLOAT Data Type Error Detection

Bug 4374847

Oracle Rdb assumes that data provided by application programs and inserted into DOUBLE PRECISION, REAL or FLOAT columns will be valid. However, there are cases when invalid floating point data was inserted into such columns (for example, data stored from uninitialized program variables).

The RMU Verify command has been enhanced with a new /VALIDATE qualifier that requests RMU Verify to scan any table containing one or more DOUBLE PRECISION, REAL or FLOAT columns and that the floating point data be checked for validity.

The following example shows the extra pass that is performed in response to the Validate qualifier.

```
$          RMU /VERIFY -
```

Oracle® Rdb for OpenVMS

```
      /VALIDATE=FLOAT=(departments,salary*) -
      /LOG -
      MF_PERSONNEL
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-DBBOUND, bound to database "DISK$TEST:[TEST_SYSTEM.DATABASE]
      MF_PERSONNEL.RDB;1"
%RMU-I-OPENAREA, opened storage area RDB$SYSTEM for protected retrieval
%RMU-I-BGNAIPVER, beginning AIP pages verification
%RMU-I-ENDAIPVER, completed AIP pages verification
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-I-OPENAREA, opened storage area MF_PERS_SEGSTR for protected retrieval
%RMU-I-ENDABMSPM, completed ABM pages verification
%RMU-I-CLOSAREAS, releasing protected retrieval lock on all storage areas
%RMU-I-BGNVALDAT, Beginning of table data validation 8-MAR-2019 15:19:09.26
%RMU-I-TABCOLVLD, Scanning table DEPARTMENTS which has 2 floating point columns.
%RMU-I-TABCOLVLD, Scanning table SALARY_HISTORY which has 1 floating point
      column.
%RMU-I-ENDVALDAT, End of data validation at 8-MAR-2019 15:19:09.31
%RMU-S-ENDVERIFY, elapsed time for verification : 0 00:00:00.27
$
```

In the previous example, the SALARY_AMOUNT column of the SALARY_HISTORY table and the BUDGET_ACTUAL and BUDGET_PROJECTED columns of the DEPARTMENTS table were modified to DOUBLE PRECISION to show the command in operation.

Usage Notes

- The Validate qualifier operates as a separate set of transactions (one per table) after other verify operations are complete. Use the Transaction_Type qualifier to alter the type of transaction being executed.
- The Validate qualifier defaults to all tables with DOUBLE PRECISION, REAL or FLOAT columns. However, the FLOAT keyword also accepts an optional list of table names to be verified. The table name may include the OpenVMS wildcard characters (*) and (%).

```
$      RMU /VERIFY -
          /VALIDATE=FLOAT=(departments,salary*) -
          /TRANSACTION_TYPE=WRITE -
          /NOLOG -
          MF_PERSONNEL
$
```

Views, global and local temporary tables, and information tables as well as misspelled names are ignored by this qualifier, even if they are named by the Validate qualifier. In such cases, no error is reported.

- If an invalid floating value is detected, then RMU will report the column name and the DBKEY of the affected row.
- The default Transaction_Type for the Validate action is PROTECTED READ mode. Use /TRANSACTION_TYPE=READ_ONLY to minimize locking.

5.1.6 New /ABORT Qualifier for the RMU/SET DATABASE Command

The new Oracle Rdb RMU SET DATABASE command /ABORT qualifier will use either the OpenVMS system service \$FORCEX or the OpenVMS system service \$DELPRC to force attached user processes out of an open Rdb database but not database server processes, such as AIJ, ROW CACHE or Database Recovery Servers, or database utility processes such as RMU. The database will remain open unless the database is an AUTOMATIC CLOSE database and no user, server or utility processes are accessing the database when the /ABORT operation completes. Currently, this functionality is only available as an option of the CLOSE command which closes the database. The /ABORT qualifier requires that the user has been granted RMU\$OPEN privilege in the root file access control list (ACL) for the database or the OpenVMS WORLD privilege.

The /ACCESS=RESTRICTED qualifier can be specified in the same RMU/SET DATABASE command as the /ABORT qualifier to require that SQL users have DBADM privilege to attach to the database. The /ACCESS=RESTRICTED qualifier will always be executed before the /ABORT qualifier.

The command line syntax for the RMU SET DATABASE command /ABORT qualifier is:

```
/ABORT [= ( FORCEX | DELPRC , [NO]CLUSTER , [NO]WAIT ) ]
```

FORCEX and NOWAIT are the defaults. If the user specifies WAIT but does not specify CLUSTER or NOCLUSTER, CLUSTER is the default. If NOWAIT is specified or defaulted to and neither CLUSTER nor NOCLUSTER is specified, the default is NOCLUSTER. If NOCLUSTER is specified, only targeted processes on the node where the SET DATABASE/ABORT command is invoked are aborted. If CLUSTER is specified, targeted users on all cluster nodes where the database is open are aborted. Therefore, if only /ABORT is specified, the default is /ABORT=(FORCEX,NOCLUSTER,NOWAIT).

If NOWAIT is specified, RMU/SET DATABASE will return the prompt to the user when the \$FORCEX or \$DELPRC requests have been issued to all of the targeted processes. If WAIT is specified, RMU/SET DATABASE will return the prompt to the user when the targeted processes are no longer accessing the database.

The FORCEX option cannot force an exit of a database process with a spawned subprocess or a suspended or swapped out process. It aborts batch jobs that are using the database. The DELPRC option deletes any subprocesses of all database users, thereby deleting the processes from the database. The DELPRC and FORCEX options are based on the OpenVMS system services \$DELPRC and \$FORCEX. Refer to the OpenVMS documentation set for more information.

In the following example, the database is open on one node and has 5 active users. The "rmu/set database/abort" command defaults to "rmu/set database/abort=(forcex/nocluster/nowait)". After this command completes, the \$FORCEX system service has been used to force these 5 users out of the database so that database maintenance can be performed. The process logs of the 5 users show the %RDMS-F-ABORTUSERS message output when a user is forced out of the database by a FORCEX operation.

```
$ rmu/show user/out=user.dat mf_personnel
$ search user.dat "active database users"
  - 5 active database users on this node
$ rmu/set database/abort mf_personnel
%RMU-I-MODIFIED, Database state modified
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
$ rmu/show user/out=user.dat mf_personnel
$ search user.dat "active database users"
%SEARCH-I-NOMATCHES, no strings matched
$ search forcex1.log abort
```

Oracle® Rdb for OpenVMS

```
%RDMS-F-ABORTUSERS, database operator requested user termination
$ search forcex2.log abort
%RDMS-F-ABORTUSERS, database operator requested user termination
$ search forcex3.log abort
%RDMS-F-ABORTUSERS, database operator requested user termination
$ search forcex4.log abort
%RDMS-F-ABORTUSERS, database operator requested user termination
$ search forcex5.log abort
%RDMS-F-ABORTUSERS, database operator requested user termination
$
```

In the following example, the database is open on two nodes and has 5 active users on each node. The "rmu/set database/abort=(delprc,cluster,wait)" command is executed. After this command completes, the \$DELPRC system service has been used to delete the processes of these 10 database users, 5 on each node, so that database maintenance can be performed. The log from the second node shows that the processes of the 5 users on the other node are no longer accessing the database after the wait time of 1 minute and 30 seconds expires, during which the SET DATABASE/ABORT command was executed on the first cluster node. The DELPRC operation deletes the user processes and therefore no abort message is output for the deleted user processes.

```
$ rmu/show user/out=user.dat mf_personnel
$ search user.dat active, node
Oracle Rdb V7.3-320 on node TEST01 16-MAY-2019 09:48:33.57
  - 5 active database users on this node
  - database is also open on the following node:
$ rmu/set database/abort=(delprc,cluster,wait) mf_personnel
%RMU-I-MODIFIED, Database state modified
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
$ rmu/show user/out=user.dat mf_personnel
$ search user.dat active, node
Oracle Rdb V7.3-320 on node TEST01 16-MAY-2019 09:49:33.95
$!
$!### BELOW IS LOG FROM THE OTHER NODE ###
$!
$ rmu/show user/out=usern.dat mf_personnel
$ search usern.dat active, node
Oracle Rdb V7.3-320 on node TEST02 16-MAY-2019 09:48:04.49
  - 5 active database users on this node
  - database is also open on the following node:
$ wait 00:01:30
$ rmu/show user/out=usern.dat mf_personnel
$ search usern.dat active, node
Oracle Rdb V7.3-320 on node TEST02 16-MAY-2019 09:49:34.55
$
```

5.1.7 Smaller After Image Backup Files

This release of Oracle Rdb reduces the size of most backup after image journal files. The size may be reduced by as much as 254 blocks compared with prior versions. This is the result of improvements made to eliminate unused padding pages in the backup.

This change only affects the backup of after image journals when compression is not used, either by specifying the /NOCOMPRESSION qualifier or when the default is no compression.

Chapter 6

Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.1

6.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.1

6.1.1 New and Modified Replication Network Error Handling Log Messages

Enhancement Bug 7282606

In order to provide more information on network error recovery procedures, some existing messages have been modified and new messages have been added to the Oracle Rdb Replication (Hot Standby) server log messages output when network failures occur while data is being transferred between a Master Rdb database and a Standby Rdb database.

- Network retry count of # exceeded
This message has been changed to:
Network retry count of # exceeded; aborting...
where # is the maximum number of times to try to re-establish the network connection. The attempt to re-establish the network connection has failed and been aborted after attempting to re-establish the connection the maximum number of times displayed in the message. The attempt to re-establish the network connection will be aborted before the retry count is reached if an attempt to shutdown and restart the network fails or an unrecoverable network error occurs during a retry attempt. The network retry count can be modified using the RDM\$BIND_HOT_NETWORK_RETRY_COUNT logical.
- Unexpected loss of network connection; retrying...
This new message has been added to indicate that the network connection has been lost and an attempt will be made to re-establish the network connection. The error status will have already been output in a previous log message.
- Unexpected loss of network connection, status: status_code – status_text; retrying...
This new message has been added to indicate that an error has been returned which indicates that the network connection has been lost and an attempt will be made to re-establish the network connection. The error status code id is given followed by the error message name and text.
- Network connection failure, server "server_name"
This message has been changed to:
Network connection failure, server "server_name"; aborting...
to indicate that the named Hot Standby server cannot connect to the network because of a network failure. The error status will have already been output in a previous log message.
- Unexpected network error status: status_code – status_text; aborting...
Unexpected network error subcode: subcode_id – subcode_text
This new message has been added to indicate that an unexpected and unrecoverable network error has been detected. The error status code id is given followed by the error message name and text. If there is a related secondary error status that status code is also given followed by the error message name and text.
- Network Connection lost, no attempt to reconnect will be made; aborting...
This new message has been added to indicate that the connection to the network has been lost but in this case any retry attempt would fail so no attempt to re-establish the network connection will be made. The error status will have already been output in a previous log message.
- Network connection re-established
This new message has been added to confirm that the network connection has been successfully

re-established and the replication operation will continue.

The following extract from the log of one of the Oracle Rdb Hot Standby servers shows two of the new messages described above, "Unexpected loss of network connection, status: 056EA018 – %COSI-W-ENDOFFILE, end of file; retrying..." and "Network connection re-established" which have been added to clarify network error recovery procedures.

```

!+++++
!
!           L C S   -   AIJ Log Catch-Up Server Process
!
!+++++

30-MAY-2018 17:47:29.47 - Reading AIJ sequence 2:97-191
30-MAY-2018 17:47:29.47 - Unexpected error LSS019: 056EC2C4 -
  %COSI-F-WRITERR, write error
30-MAY-2018 17:47:29.47 - Unexpected subcode: 000020E4 -
  %SYSTEM-F-LINKABORT, network partner aborted logical link
30-MAY-2018 17:47:29.47 - Unexpected loss of network connection,
status: 056EA018 - %COSI-W-ENDOFFILE, end of file; retrying...
30-MAY-2018 17:47:29.47 - Pausing for 1 second before retrying connect
30-MAY-2018 17:47:30.47 - Attempting to re-establish connection...
30-MAY-2018 17:47:30.47 - Connecting to node "TESTER"
30-MAY-2018 17:47:30.47 - Service object name is "RDMAIJ731"
30-MAY-2018 17:47:30.49 - Network protocol is DECnet
30-MAY-2018 17:47:30.49 - Identified standby LRS process 209E1B3B
30-MAY-2018 17:47:30.49 - Network connection re-established

```

6.1.2 RMU Show Statistics Feature To Select Screens To Include In STATISTICS.RPT

Enhancement Bug 8945775

The RMU Show Statistics command allows the /WRITE_REPORT_DELAY=n qualifier to collect statistics for "n" seconds and then exit after writing a text file named STATISTICS.RPT that contains the current statistics for all of the RMU Show Statistics screens.

With this release of Oracle Rdb, RMU allows individual screens to be included or excluded from the STATISTICS.RPT file by specifying a full screen name or parts of a screen name using the new REPORT_SCREEN qualifier. The specification may include wild card character "*" to match multiple characters and "%" to match a single character.

The new RMU Show Statistic qualifier /REPORT_SCREEN supports these keywords:

- INCLUDE="string"
/REPORT_SCREEN=(INCLUDE="string")
This keyword will include in the report only those screens with a screen name matching all or part of the specified "string". The string may contain wild card characters.
- EXCLUDE="string"
/REPORT_SCREEN=(EXCLUDE="string")
This keyword will exclude from the report only those screens with a screen name matching all or part of the specified "string". The string may contain wild card characters.

INCLUDE and EXCLUDE options cannot both be specified in the same RMU Show Statistics command. The REPORT_SCREEN qualifier is only valid when the WRITE_REPORT_DELAY qualifier is used.

To include or exclude all the "SUMMARY" screens from the report, the commands would be similar to these examples.

```
$ RMU/SHOW STATISTICS -
  /WRITE_REPORT_DELAY=120 -
  /REPORT_SCREEN=INCLUDE="SUMMARY*" -
  DATABASE.RDB
$
$
$ RMU/SHOW STATISTICS -
  /WRITE_REPORT_DELAY=120 -
  /REPORT_SCREEN=EXCLUDE="Summary*" -
  DATABASE.RDB
$
```

The string matching is case insensitive. That is, the strings "SUMMARY*" and "Summary*" will be treated the same and will match the following screen header titles containing "Summary".

- Summary IO Statistics
- Summary Locking Statistics
- Summary Object Statistics
- Summary Tx Statistics

An example of an RMU/SHOW STATISTICS screen header containing the unique screen name "Summary IO Statistics" is:

```
Node: NODNAM (1/1/16)                Oracle Rdb V7.3-31
Perf. Monitor                        24-MAY-2018 15:14:32.85
Rate: 3.00 Seconds                   Summary IO Statistics
Page: 1 of 1 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 Mode: Online
```

6.1.3 RMU BACKUP and RESTORE /OUTPUT Qualifier to Write Output to a File

Enhancement Bug 1108555

An "/OUTPUT=file-spec" qualifier, which redirects log output (which can be voluminous), from SYS\$OUTPUT to the specified file, was added to the Oracle Rdb RMU BACKUP and RESTORE commands in the Rdb V7.2-310 release but was not documented in the sections of the Oracle RMU Reference Manual describing these commands. This documentation will be added to the next version of the RMU Reference Manual.

The syntax of the OUTPUT qualifier is:

```
/OUTPUT=file-spec
```

If a file extension is not specified in the output file specification, the output file extension will be ".LIS".

Oracle® Rdb for OpenVMS

The default if this qualifier is not specified is to send output from the BACKUP or RESTORE command to SY\$OUTPUT. This qualifier must specify a valid file specification and cannot be negated. The /NOLOG qualifier cannot be specified on the same command line as the /OUTPUT qualifier. The qualifier /LOG=BRIEF or /LOG=FULL can be specified with the /OUTPUT=file_spec qualifier to control the amount of information written to the designated output file. The /LOG=BRIEF (the default) qualifier will display the start and completion time of the backup or restore of each database storage area. The /LOG=FULL qualifier will also display thread assignment information and statistical information for each storage area. See the Oracle RMU Reference Manual for more information on the use of the /LOG qualifier with the BACKUP and RESTORE commands.

In the following example, the /LOG=BRIEF qualifier is used first with the RMU/RESTORE command and then with the RMU/BACKUP command to direct output to the TEST.LIS file.

```
$ rmu/backup/log=brief/output=test mf_personnel mfp.rbf
$ dir test.lis

Directory DISK:[DIRECTORY]

TEST.LIS;1

Total of 1 file.
$ search test.lis RMU-I-COMPLETED
%RMU-I-COMPLETED, BACKUP operation completed at 27-JUN-2018 16:41:00.35
$ delete test.lis;*
$!
$ sql
drop database file disk:[directory]mf_personnel;
$ rmu/restore/nocdd/dir=disk:[directory]/log=brief/output=test mfp.rbf
$ dir test.lis
Directory DISK:[DIRECTORY]

TEST.LIS;1

Total of 1 file.
$ search test.lis RMU-I-COMPLETED
%RMU-I-COMPLETED, RESTORE operation completed at 27-JUN-2018 16:43:00.37
```

In the following example, the /LOG=FULL qualifier is used first with the RMU/RESTORE command and then with the RMU/BACKUP command to direct output to the TEST.LIS file.

```
$ rmu/backup/log=full/output=test mf_personnel mfp.rbf
$ dir test.lis

Directory DISK:[DIRECTORY]

TEST.LIS;1

Total of 1 file.
$ search test.lis RMU-I-COMPLETED
%RMU-I-COMPLETED, BACKUP operation completed at 27-JUN-2018 16:45:00.23
$ delete test.lis;*
$!
$ sql
drop database file disk:[directory]mf_personnel;
$ rmu/restore/nocdd/dir=disk:[directory]/log=full/output=test mfp.rbf
$ dir test.lis

Directory DISK:[DIRECTORY]
```

```
TEST.LIS;1
```

```
Total of 1 file.
```

```
$ search test.lis RMU-I-COMPLETED
```

```
%RMU-I-COMPLETED, RESTORE operation completed at 27-JUN-2018 16:48:00.22
```

6.1.4 AIJ Backup Data Compression Information is Now in the ABS Process Logs

New messages have been added to the Oracle Rdb After Image Journal Automatic Backup Server process logs which will be output if data compression is enabled for automatic database AIJ backups. Automatic database AIJ backups using data compression can be defined for a database by the RMU/SET AFTER_JOURNAL command. A log file to be output by each automatic AIJ backup server process can be enabled by the RMU/SET SERVER ABS command for a database or by defining the system RDM\$BIND_ABS_LOG_FILE logical for a cluster node.

The new automatic AIJ backup server process log messages output if data compression is enabled for database automatic AIJ backup files are the following.

- Compression ZLIB level 6
This first message shows the currently supported data compression algorithm "ZLIB" and the ZLIB level used, which will be an integer between 1 and 9. The higher the level number, the greater the compression but also the greater amount of CPU time spent doing the compression. The default level of 6 is a good trade off between the necessary CPU time and the amount of data compression.
- Data compressed by 39% (3956 KB in/2428 KB out)
This second compression message, which immediately follows the first compression message, shows the amount of compression as a percent value based on the total number of input uncompressed bytes compared to the total number of output compressed bytes followed by the total number of uncompressed bytes and the total number of compressed bytes expressed in scaling units, which will vary depending on the amount of data compressed. The scaling units will be one of "Bytes", "KB" for kilobytes, "MB" for megabytes, "GB" for gigabytes or "TB" for Terabytes.

The following example shows the last portion of a log file named ABS_23CF6459.OUT created by an automatic backup server process with a process id of 23CF6459. The log messages show that this process has created the After Image Journal backup file AIJBCKCOMP.ABF;24 and that the data in the backup file was compressed using the ZLIB compression level 6 algorithm. The compressed output data was 38% smaller than the uncompressed input data. The number of uncompressed input kilobytes is 3932 and the number of compressed output kilobytes in the AIJBCKCOMP.ABF;24 output AIJ backup file is 2445.

```
$ TYPE DEVICE:[DIRECTORY]ABS_23CF6459.OUT
```

```
4-OCT-2018 08:44:33.45 - AIJ Backup Server (ABS) activated
4-OCT-2018 08:44:33.45 - Database is DEVICE:[DIRECTORY]AIJBCKCOMP.RDB;1
4-OCT-2018 08:44:33.45 - Backing up AIJ 23
4-OCT-2018 08:44:33.45 - By-sequence AIJ backup for sequence 23 to 23
%RDMS-I-OPERNOTIFY, system operator notification: AIJ backup operation started
4-OCT-2018 08:44:33.45 - No Hot Standby servers active
%RDMS-I-AIJBCKSEQ, backing up after-image journal sequence number 23
%RDMS-I-LOGBCKAIJ, backing up after-image journal J4 at 08:44:33.45
%RDMS-I-LOGCREBCK, created backup file DEVICE:[DIRECTORY]AIJBCKCOMP.ABF;24
%RDMS-I-OPERNOTIFY, system operator notification: AIJ backup operation completed
```

```

4-OCT-2018 08:44:34.34 - AIJ backup complete
4-OCT-2018 08:44:34.34 - Compression ZLIB level 6
4-OCT-2018 08:44:34.34 - Data compressed by 38% (3932 KB in/2445 KB out)
4-OCT-2018 08:44:34.34 - ELAPSED:      0 00:00:00.91  CPU: 0:00:00.79  BUFIO:
20 DIRIO: 356  FAULTS: 242
$

```

6.1.5 RMU MOVE_AREA and COPY_DATABASE /OUTPUT Qualifier to Write Output to a File

A new "/OUTPUT=file-spec" qualifier has been added to the Oracle Rdb RMU MOVE_AREA and COPY_DATABASE commands. This qualifier will redirect log output, which can be voluminous, from SYSS\$OUTPUT to the specified file.

The syntax of the OUTPUT qualifier is:

```
/OUTPUT=file-spec
```

If a file extension is not specified in the output file specification, the output file extension will be ".LIS".

The default if this qualifier is not specified is to send output from the MOVE_AREA or COPY_DATABASE command to SYSS\$OUTPUT. This qualifier must specify a valid file specification and cannot be negated. The /NOLOG qualifier cannot be specified on the same command line as the /OUTPUT qualifier. See the Oracle RMU Reference Manual for more information on the use of the /LOG qualifier with the MOVE_AREA and COPY_DATABASE commands.

In the following example, the /LOG and /OUTPUT qualifiers are specified with the RMU/MOVE_AREA command to direct command output to the MOVELOG.LIS file when moving all MF_PERSONNEL database areas to a different directory.

```

$ RMU/MOVE_AREA/LOG/OUTPUT=MOVELOG MF_PERSONNEL -
  /ALL_AREAS/DIRECTORY=DISK:[DIRECTORY]
$ SEAR MOVELOG.LIS MOVTXT_15, MOVTXT_06, MOVTXT_07, DOFULLBCK,-
  "operation completed"
%RMU-I-MOVTXT_15, Area files have been moved
%RMU-I-MOVTXT_06, Database root updated for all moved areas
%RMU-I-MOVTXT_07, Obsolete files deleted
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
%RMU-I-COMPLETED, MOVE_AREA operation completed at 18-JUL-2018 16:53:00.37

```

In the following example, the /LOG and /OUTPUT qualifiers are specified with the RMU/COPY_DATABASE command to direct command output to the COPYLOG.LIS file when copying the MF_PERSONNEL database to a different directory.

```

$ RMU/COPY_DATABASE/LOG/OUTPUT=COPYLOG MF_PERSONNEL /DIRECTORY=DISK:[DIRECTORY]
$ SEAR COPYLOG.LIS MOVTXT_00, DOFULLBCK, "operation completed"
%RMU-I-MOVTXT_00, Moved root file DISK:[DIRECTORY]MF_PERSONNEL.RDB;VERSION
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
%RMU-I-COMPLETED, COPY_DATABASE operation completed at 18-JUL-2018 17:23:00.24

```

6.1.6 Support for /OUTPUT=file-spec Qualifier Added to RMU Backup Plan File

Enhancement Bug 1108555

The Oracle Rdb RMU/BACKUP command "/OUTPUT=file-spec" qualifier, which redirects log output, which can be voluminous, from SYS\$OUTPUT to the specified file, was not supported for the RMU Backup Plan file and therefore could not be executed by the RMU/BACKUP/PLAN command even though it could be specified for the RMU/BACKUP command. Support for the /OUTPUT=file-spec qualifier has now been added to the RMU Backup Plan file used for parallel backups of Rdb databases.

The command line syntax of the OUTPUT qualifier for the RMU Backup command line is:

```
/OUTPUT=file-spec
```

The new syntax for specifying the OUTPUT qualifier in the RMU Backup Plan file is:

```
/OUTPUT = file-spec
```

The placeholder for the OUTPUT qualifier in the RMU Backup Plan file is:

```
! Output = specification for output file
```

The default if this qualifier is not specified is to send output from the BACKUP command to SYS\$OUTPUT. If a file extension is not specified in the output file specification, the output file extension will be ".LIS". This qualifier must specify a valid file specification and cannot be negated.

In this example, the first RMU/BACKUP/PARALLEL/OUTPUT=PARALLEL_LOG command executes a parallel backup to disk logging output to the PARALLEL_LOG.LIS;1 log file and creates the MFP.PLAN file which contains the "Output = PARALLEL_LOG" qualifier and the other qualifiers specified in the backup command. Then the second RMU/BACKUP/PLAN MFP.PLAN command is executed to repeat the same backup using the same command qualifiers as the first command which have been saved by the first command in the MFP.PLAN file. For both commands, the parallel backup output has been logged to a file named PARALLEL_LOG.LIS, producing the files PARALLEL_LOG.LIS;1 and PARALLEL_LOG.LIS;2.

```
$ rmu/backup/parallel=executor=1/disk/execute/log -
  /output=parallel_log -
  /active_io=5 -
  /page_buffers=5 -
  /noquiet_point -
  /list_plan=mfp.plan -
  mf_personnel [.test1]mfp.rbf,[.test2],[.test3]
$
$ sear mfp.plan "output"
  Output = PARALLEL_LOG
$
$ rmu/backup/plan mfp.plan
$
$ type parallel_log.lis;2
WORKER_001: %RMU-I-BCKTXT_00, Backed up root file
DISK:[DIRECTORY]MF_PERSONNEL.RDB;1
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area (RDB$SYSTEM)
DISK:[DIRECTORY]MF_PERS_DEFAULT.RDA;1 at 3-AUG-2018 15:03:39.44
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area (RDB$SYSTEM)
```

Oracle® Rdb for OpenVMS

```
DISK:[DIRECTORY]MF_PERS_DEFAULT.RDA;1 at 3-AUG-2018 15:03:39.45
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area (EMPIDS_MID)
DISK:[DIRECTORY]EMPIDS_MID.RDA;1 at 3-AUG-2018 15:03:39.45
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area
(MF_PERS_SEGSTR)
DISK:[DIRECTORY]MF_PERS_SEGSTR.RDA;1 at 3-AUG-2018 15:03:39.45
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area (EMPIDS_MID)
DISK:[DIRECTORY]EMPIDS_MID.RDA;1 at 3-AUG-2018 15:03:39.46
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area
(MF_PERS_SEGSTR)
DISK:[DIRECTORY]MF_PERS_SEGSTR.RDA;1 at 3-AUG-2018 15:03:39.46
WORKER_001: %RMU-I-RESUME, resuming operation on volume 2 using DISK
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area
(SALARY_HISTORY)
DISK:[DIRECTORY]SALARY_HISTORY.RDA;1 at 3-AUG-2018 15:03:39.47
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area (EMP_INFO)
DISK:[DIRECTORY]EMP_INFO.RDA;1 at 3-AUG-2018 15:03:39.47
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area (EMPIDS_LOW)
DISK:[DIRECTORY]EMPIDS_LOW.RDA;1 at 3-AUG-2018 15:03:39.47
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area
(SALARY_HISTORY)
DISK:[DIRECTORY]SALARY_HISTORY.RDA;1 at 3-AUG-2018 15:03:39.48
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area (EMP_INFO)
DISK:[DIRECTORY]EMP_INFO.RDA;1 at 3-AUG-2018 15:03:39.48
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area (EMPIDS_LOW)
DISK:[DIRECTORY]EMPIDS_LOW.RDA;1 at 3-AUG-2018 15:03:39.48
WORKER_001: %RMU-I-RESUME, resuming operation on volume 3 using DISK
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area (JOBS)
DISK:[DIRECTORY]JOBS.RDA;1 at 3-AUG-2018 15:03:39.49
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area (EMPIDS_OVER)
DISK:[DIRECTORY]EMPIDS_OVER.RDA;1 at 3-AUG-2018 15:03:39.49
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area (DEPARTMENTS)
DISK:[DIRECTORY]DEPARTMENTS.RDA;1 at 3-AUG-2018 15:03:39.49
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area (JOBS)
DISK:[DIRECTORY]JOBS.RDA;1 at 3-AUG-2018 15:03:39.50
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area
(EMPIDS_OVER)
DISK:[DIRECTORY]EMPIDS_OVER.RDA;1 at 3-AUG-2018 15:03:39.50
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area
(DEPARTMENTS)
DISK:[DIRECTORY]DEPARTMENTS.RDA;1 at 3-AUG-2018 15:03:39.50
WORKER_001: %RMU-I-COMPLETED, BACKUP operation completed at
3-AUG-2018 15:03:39.51
%RMU-I-COMPLETED, BACKUP operation completed at 3-AUG-2018 15:03:39.51
$
$ directory parallel_log.lis;*

Directory DISK:[DIRECTORY]

PARALLEL_LOG.LIS;2 PARALLEL_LOG.LIS;1

Total of 2 files.
$
```

6.1.7 New CREATE OR REPLACE Support for PROFILE

This release of Oracle Rdb supports the CREATE OR REPLACE syntax for PROFILE. If the named profile does not exist, then CREATE OR REPLACE acts like a CREATE PROFILE statement. If the named profile exists, it is first deleted and then replaced by the new definition.

Any dependencies upon the profile, for example being assigned to a database USER, are not affected by the CREATE OR REPLACE PROFILE statement.

The following example shows a profile named ADMIN_USER which is replaced by a new definition. It also demonstrates that the assignment to a user is unchanged.

```
SQL> show profiles admin_user;
ADMIN_USER
Transaction modes (shared read, no batch update)
Default transaction read write wait 3
Isolation level read committed
SQL>
SQL> create user rdbuser1
cont>     identified externally
cont>     profile admin_user
cont> ;
SQL>
SQL> show user rdbuser1;
RDBUSER1
Identified externally
Account is unlocked
Profile: ADMIN_USER
No roles have been granted to this user
SQL>
SQL> create or replace profile admin_user
cont>     transaction modes (shared read, shared write, no batch update)
cont> ;
SQL>
SQL> show profiles admin_user;
ADMIN_USER
Transaction modes (shared, no batch update)
SQL>
SQL> show user rdbuser1;
RDBUSER1
Identified externally
Account is unlocked
Profile: ADMIN_USER
No roles have been granted to this user
SQL>
```

The default profile, created with the CREATE DEFAULT PROFILE statement, can also be replaced.

```
SQL> create or replace default profile
cont>     default transaction
cont>     read write
cont>     isolation level read committed
cont>     limit rows 10000
cont> ;
SQL>
```

6.1.8 New CREATE OR REPLACE Support for VIEW

Enhancement Bugs 3763080 and 4555512

This release of Oracle Rdb now supports the CREATE OR REPLACE VIEW Statement. The OR REPLACE modifier to CREATE VIEW will request that SQL replace the view if one of that name exists in the database.

Arguments

- OR REPLACE

This clause instructs SQL to replace an existing view if possible. If the view does not exist, it will be created. The restrictions upon the replace action are listed in the usage notes.

Usage Notes

- If the view does not exist, then there must not be a table, sequence or synonym with the same name as this new view.
- If the view exists and the CREATE VIEW statement was used, then an error will be reported.
- If the view exists and the CREATE OR REPLACE VIEW statement was used and the name used is a synonym, then the view referenced by that synonym will be replaced.
- A view will be replaced if these conditions are met.
 1. There are no existing database object dependencies on the view.
For example, there are no procedures, functions or other objects with references to the view and its columns.

Note

Dependencies may exist externally such as SQL Pre-compiler source code or SQL Module Language procedures. Replacing the view with an incompatible version may cause those modules to execute in unexpected ways or to generate errors when recompiled.

2. The existing dependencies are met by the new view definition.
For example, if a view is referenced by a stored procedure then any column names referenced must exist after the replace is complete.
Consider this example which attempts to reduce the columns of the view. The view CURRENT_INFO uses the view field SALARY_START, which is no longer present in the revised view definition.

```
SQL> create or replace view CURRENT_SALARY
cont>   (LAST_NAME,
cont>    FIRST_NAME,
cont>    EMPLOYEE_ID,
cont>    SALARY_AMOUNT) as
cont>   (select
cont>      C2.LAST_NAME,
cont>      C2.FIRST_NAME,
cont>      C2.EMPLOYEE_ID,
cont>      C1.SALARY_AMOUNT
cont>   from SALARY_HISTORY C1, EMPLOYEES C2
cont>   where ((C1.SALARY_END is null)
cont>          and (C2.EMPLOYEE_ID = C1.EMPLOYEE_ID)));
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-VIEWDEPEND, other database objects are dependent on this view
-RDMS-F-NOCHGVW, the definition of view "CURRENT_SALARY" may not be changed
SQL>
```

3. There are no language semantic requirements due to the functionality used to reference the view.
For example, an INSERT INTO statement that omits the column list or a SELECT * FROM statement have an implied column name list and also column ordering. This column name

ordering must be maintained by the replace.

Note

*The EXISTS function allows the format EXISTS (SELECT * FROM ... WHERE ...). However, using the * syntax in this context does not actually expand to reference all columns and therefore is not considered as a semantic restriction.*

- Oracle Rdb does not check for compatible data types so it is possible that functions, procedures, and views may fail due to incompatible types. In some cases, the ALTER MODULE ... COMPILE statement should be used to validate such changes prior to committing the replacement of the view.
- If the view exists and is replaced, then any column level comment, granted access control or audit settings will be propagated to the replacement view if the column name is the same as in the prior version.
- If the view exists and is replaced, then any view comment, granted access control or audit/alarm settings will be propagated to the replacement view.
- If the view exists and the OR REPLACE clause is used, then you must have ALTER privilege on the referenced view.

Examples

Example 1: This example shows a definition of CURRENT_INFO that can be applied to the PERSONNEL database when the view exists or even when the view does not yet exist.

```
SQL> create or replace view CURRENT_INFO
cont>     (LAST_NAME,
cont>     FIRST_NAME,
cont>     "ID",
cont>     DEPARTMENT,
cont>     JOB,
cont>     JSTART,
cont>     SSTART,
cont>     SALARY) as
cont>     (select
cont>         C1.LAST_NAME,
cont>         C1.FIRST_NAME,
cont>         C1.EMPLOYEE_ID,
cont>         C2.DEPARTMENT_NAME,
cont>         C3.JOB_TITLE,
cont>         C1.JOB_START,
cont>         C4.SALARY_START,
cont>         C4.SALARY_AMOUNT
cont>     from CURRENT_JOB C1, DEPARTMENTS C2, JOBS C3, CURRENT_SALARY C4
cont>     where (((C2.DEPARTMENT_CODE = C1.DEPARTMENT_CODE)
cont>         and (C3.JOB_CODE = C1.JOB_CODE))
cont>         and (C4.EMPLOYEE_ID = C1.EMPLOYEE_ID)));
SQL>
```

6.1.9 New CREATE OR REPLACE Support for SEQUENCE

Enhancement Bug 3763080

This release of Oracle Rdb now supports the CREATE OR REPLACE SEQUENCE Statement. The OR REPLACE modifier to CREATE SEQUENCE will request that SQL replace the sequence if one of that name exists in the database.

Arguments

◆ OR REPLACE

This clause instructs SQL to replace an existing sequence if possible. If the sequence does not exist, it will be created. The restrictions upon the replace action are listed in the usage notes.

Usage Notes

- ◆ You must have the CREATE database privilege to execute the CREATE SEQUENCE Statement. You must have the ALTER sequence privilege to execute the CREATE OR REPLACE SEQUENCE Statement on an existing sequence.
- ◆ If the sequence does not exist, then there must not be a table, synonym or view with the same name as this new sequence.
- ◆ If the sequence exists and the CREATE SEQUENCE Statement was used, then an error will be reported.
- ◆ If the sequence exists and the CREATE OR REPLACE SEQUENCE Statement was used and the name used is a synonym, then the sequence referenced by that synonym will be replaced.
- ◆ A system sequence (created internally by Oracle Rdb), or column identity sequence (created by the IDENTITY or GENERATED ... AS IDENTITY clause), may not be replaced.
- ◆ If a sequence is replaced, the START WITH value will be reset to the value specified by the CREATE SEQUENCE Statement or the default based on the other clauses in the statement.
- ◆ Exclusive access is required if a sequence is replaced. No active queries or other users may have access to that sequence.
- ◆ The access control list (ACL) on the sequence is propagated (saved and restored) from the old sequence by the REPLACE action.
- ◆ The audit and alarm settings for the sequence are propagated (saved and restored) from the old sequence by the REPLACE action.
- ◆ Any comment on the sequence is propagated (saved and restored) from the old sequence by the REPLACE action unless there is a COMMENT IS clause specified by the CREATE OR REPLACE Statement. Comments can be created by the COMMENT IS clause of the CREATE or ALTER SEQUENCE Statement or by the COMMENT ON SEQUENCE Statement.

Examples

Example 1: This example shows the CREATE OR REPLACE SEQUENCE Statement and demonstrates that any comment or access control list is propagated by OR REPLACE action.

```
create or replace sequence DEPT_ID
  cycle noorder
  start with 10
  default wait
;

-- show that comment and ACL are propagated by OR REPLACE
show sequence DEPT_ID;
  DEPT_ID
Sequence Id: 3
Initial Value: 10
```

```

Minimum Value: 1
Maximum Value: 9223372036854775806
Next Sequence Value: 10
Increment by: 1
Cache Size: 20
No Order
Cycle
No Randomize
Comment:          revised; new departments get a unique number
    
```

```

show protection on sequence DEPT_ID;
Protection on Sequence DEPT_ID
  (IDENTIFIER=[ACCT,ACCT_USER],ACCESS=SELECT)
  (IDENTIFIER=[ACCT,ACCTUSER2],ACCESS=SELECT+SHOW+ALTER+DROP+DBCTRL
+REFERENCES)
  (IDENTIFIER=[ACCT,ACCTUSER1],ACCESS=SELECT+SHOW+ALTER+DROP+DBCTRL
+REFERENCES)
  (IDENTIFIER=[*,*],ACCESS=NONE)
    
```

6.1.10 New Options for SET LOGFILE Statement

This release of Oracle Rdb adds new options to the SET LOGFILE statement.

◆ LOGFILE quoted-filespec

This statement allows the executing SQL script to save output to an OpenVMS file. Output and errors from interactive SQL, as well as those statements, will be written to the file-spec specified.

The SET LOGFILE is functionally equivalent to the SET OUTPUT statement. A SET LOGFILE command that does not specify a file is equivalent to SET NOLOGFILE.

Various keywords can be used to control the written output file.

◇ CACHE

This is the default. The OpenVMS file caching will be in effect for this file.

◇ NOCACHE

This option disables the OpenVMS file caching for this file. Use this to prevent unnecessary caching for a temporary file.

◇ ECHO

This is the default. As well as writing the output to the designated file, all commands and errors generated by interactive SQL are also written to SYSS\$OUTPUT.

◇ NOECHO

If the option NOECHO is used, output to SYSS\$OUTPUT is disabled. All commands and errors generated by interactive SQL are only written to the output file.

◇ LARGE_FILE

If the output written to the LOGFILE is lengthy (such as when capturing the output from a query), then this option will use an RMS EXTENT size of 8192. This might improve output performance for very large files.

◇ SHARED

The file is created with the shared attribute which will allow other processes to open and read that file while it is being written by SQL.

6.1.11 New UNDECLARE CURSOR Statement

This release of Oracle Rdb introduces a new UNDECLARE CURSOR statement for interactive SQL.

This statement implicitly closes the named cursor, removes the declared cursor name from the known cursor list, and releases resources held by SQL and the Oracle Rdb Server for that cursor. If this is a table cursor, then all associated list cursors are also undeclared.

Environment

You can use the UNDECLARE CURSOR statement:

- ◆ In interactive SQL

Format

```
UNDECLARE CURSOR <cursor-name>
```

Arguments

cursor-name

Specifies the name of the declared cursor.

Example

Example of using the Undeclare Cursor statement.

```
SQL> declare mycursor cursor for select * from mytable;
SQL>
SQL> open mycursor;
SQL> fetch mycursor;
  MYFIELD
  SAMPLE
SQL> close mycursor;
SQL>
SQL> undeclare cursor mycursor;
SQL>
```

6.1.12 Enhanced LIKE Table Support in CREATE TABLE Statement

This release of Oracle Rdb introduces support for the ANSI and ISO SQL Language Standard syntax for the LIKE table clause.

In prior releases of Oracle Rdb, a table can be created using syntax similar to the following:

```
SQL> create table RETIRED_EMPLOYEES
cont>      like EMPLOYEES
cont>      ;
SQL>
```

Oracle® Rdb for OpenVMS

This statement copies the definitions of each column as well as DEFAULT values defined for those source columns. SQL also allows additional columns and constraints to be defined for the new table.

```
SQL> create table RETIRED_EMPLOYEES
cont>     like EMPLOYEES
cont>     (retirement_date DATE
cont>     ,check (retirement_date > birthday) not deferrable
cont>     );
SQL>
```

This syntax is retained for backward compatibility with prior releases of Oracle Rdb.

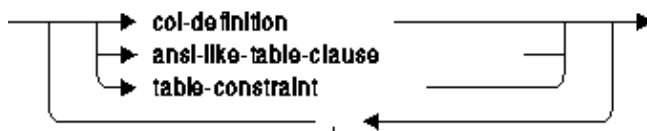
The syntax for a similar feature in the ANSI/ISO SQL Database Language moves the LIKE clause into the section that defines the columns and constraint. This adds the ability to copy column definitions from more than one table, control how GENERATED, AUTOMATIC, IDENTITY and COMPUTED columns are inherited, as well as define the column ordering; this is determined by the order of the listed columns and tables.

```
SQL> create table RETIRED_EMPLOYEES
cont>     (retirement_date DATE
cont>     ,like EMPLOYEES
cont>     including COMPUTED
cont>     excluding DEFAULTS
cont>     ,check (retirement_date > birthday) not deferrable
cont>     ,unique (employee_id)
cont>     ,hr_authorizations LIST OF BYTE VARYING
cont>     );
SQL>
```

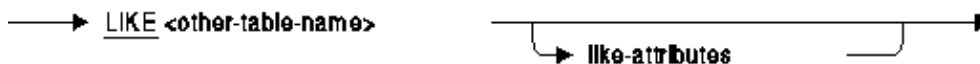
By default, GENERATED, AUTOMATIC, IDENTITY and COMPUTED columns are not copied but columns representing the same data types are created instead.

Syntax

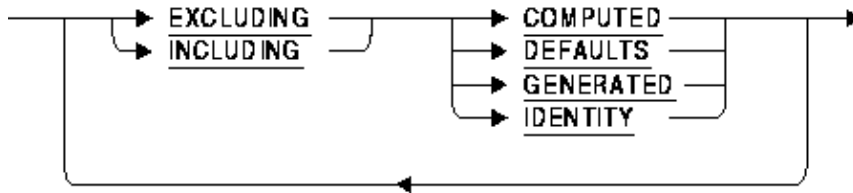
column-constraint-list =



ansi-like-table-clause =



like-attributes =



Usage Notes

- ◆ When using the LIKE clause to copy a table definition, the creator of the new table must have REFERENCES or SELECT privilege granted for the referenced table.
- ◆ The LIKE clause can be used multiple times within a CREATE TABLE statement. However, if the copied tables include any duplicate column names, then an error will be reported. Only one IDENTITY column can be defined or inherited. Use the INCLUDING IDENTITY clause, if necessary, to inherit the attributes from the referenced table. The default behavior is EXCLUDING COMPUTED, GENERATED, IDENTITY column details. In this case, non-generated columns will be created which contain the same data type attributes. Default values defined for the source tables are not automatically inherited. Use the INCLUDING DEFAULTS clause to control this behavior.
Note: For backward compatibility with previous versions of Oracle Rdb, the LIKE clause used outside the column-constraint-list defaults to INCLUDING GENERATED, INCLUDING IDENTITY, INCLUDING COMPUTED and INCLUDING DEFAULTS. The like-attributes may not be specified in this location and therefore these defaults may not be changed.
- ◆ The clauses EXCLUDING GENERATED or INCLUDING GENERATED apply to columns defined using the GENERATED ... AS (expr) and AUTOMATIC ... AS (expr) syntax. When EXCLUDING is used or implied, the generated (or automatic) column is converted to a simple base column with the same data types.
- ◆ The clauses EXCLUDING IDENTITY or INCLUDING IDENTITY apply to columns defined using the GENERATED ... AS IDENTITY and IDENTITY (...) syntax. When EXCLUDING is used or implied, the identity column is converted to a simple base column with the same data types.
- ◆ The clauses EXCLUDING COMPUTED or INCLUDING COMPUTED apply to columns defined using the COMPUTED BY expr syntax. When EXCLUDING is used or implied, the computed by column is converted to a simple base column with the same data types. Note that the column will require space in the defined table, which isn't true for COMPUTED BY columns.
- ◆ When the LIKE clause is used within the column-constraint-list, then EXCLUDING DEFAULTS is assumed. Use the INCLUDING DEFAULTS if you wish the inherited columns to have DEFAULTS inherited from the source table.
- ◆ The LIKE clause is only used to inherit the column definitions from the referenced table. Once the table is created with LIKE clauses, subsequent changes to the source table are not propagated to the created tables.

Examples

The following example shows the use of the LIKE clause to inherit columns from various template

tables.

```

SQL> create table NAMES_REC
cont>     (LAST_NAME          LAST_NAME_DOM
cont>     ,FIRST_NAME         FIRST_NAME_DOM
cont>     ,MIDDLE_INITIAL     MIDDLE_INITIAL_DOM
cont>     );
SQL>
SQL> create table ADDRESS_REC
cont>     (ADDRESS_DATA_1     ADDRESS_DATA_1_DOM
cont>     ,ADDRESS_DATA_2     ADDRESS_DATA_2_DOM
cont>     ,CITY                 CITY_DOM
cont>     ,STATE                STATE_DOM
cont>     ,POSTAL_CODE         POSTAL_CODE_DOM
cont>     );
SQL>
SQL> create table employees
cont>     (EMPLOYEE_ID        ID_DOM not null
cont>     ,like NAMES_REC      including DEFAULTS
cont>     ,like ADDRESS_REC    including DEFAULTS
cont>     ,SEX                  SEX_DOM
cont>     ,BIRTHDAY             DATE_DOM
cont>     ,STATUS_CODE         STATUS_CODE_DOM
cont>     );
SQL>

```

The resulting CREATE TABLE for the EMPLOYEES table is easier to read and allows for consistency among similar definitions.

```

SQL> show table (column) EMPLOYEES;
Information for table EMPLOYEES

```

Columns for table EMPLOYEES:

Column Name	Data Type	Domain
EMPLOYEE_ID	CHAR(5)	ID_DOM
Not Null constraint EMPLOYEES_EMPLOYEE_ID_NOT_NULL		
LAST_NAME	CHAR(14)	LAST_NAME_DOM
FIRST_NAME	CHAR(10)	FIRST_NAME_DOM
MIDDLE_INITIAL	CHAR(1)	MIDDLE_INITIAL_DOM
ADDRESS_DATA_1	CHAR(25)	ADDRESS_DATA_1_DOM
ADDRESS_DATA_2	CHAR(20)	ADDRESS_DATA_2_DOM
CITY	CHAR(20)	CITY_DOM
STATE	CHAR(2)	STATE_DOM
POSTAL_CODE	CHAR(5)	POSTAL_CODE_DOM
SEX	CHAR(1)	SEX_DOM
BIRTHDAY	DATE VMS	DATE_DOM
STATUS_CODE	CHAR(1)	STATUS_CODE_DOM

```

SQL>

```

6.1.13 New TO_DSINTERVAL and TO_YMINTERVAL Functions

This release of Oracle Rdb supports two new builtin functions: TO_DSINTERVAL and TO_YMINTERVAL. These functions accept a character string containing either an ANSI/ISO SQL Language interval format string or an ISO durations string (see below for a description).

Additionally, these functions support an optional DEFAULT ... ON CONVERSION ERROR clause which provides an alternate value to use if the source string contains errors.

The following example shows the use of the DEFAULT clause to allow special handling of the unexpected value.

```
SQL> begin
cont> declare :v interval year to month;
cont> declare :SQLSTATE_DATA_INV_PARAM constant char(5) = '22023';
cont>
cont> set :v = to_ymininterval (:duration default '-P99999Y' on conversion error);
cont> if :v = interval'-99999-0' year (5) to month
cont> then
cont>     signal :SQLSTATE_DATA_INV_PARAM ('duration conversion error');
cont> end if;
cont> end;
%RDB-E-SIGNAL_SQLSTATE, routine "(unnamed)" signaled SQLSTATE "22023"
-RDB-I-TEXT, duration conversion error
SQL>
```

Syntax

```
TO_DSINTERVAL ( ds_duration_string
                [ DEFAULT ds_return_value ON CONVERSION ERROR ]
              )

TO_YMININTERVAL ( ym_duration_string
                 [ DEFAULT ym_return_value ON CONVERSION ERROR ]
               )
```

The arguments to these functions are string values; variables, literals or function results. For TO_DSINTERVAL, ds_duration_string and ds_return_value must be formatted as a date/time DAY TO SECOND intervals. For TO_YMININTERVAL, ym_duration_string and ym_return_value must be formatted as a date/time YEAR TO MONTH intervals.

If ds_duration_string or ym_duration_string results in a NULL value, then the result of the function will be NULL. If a conversion error occurs and the DEFAULT ... ON CONVERSION ERROR value results in a NULL value, then the result of the function will be NULL.

The clause DEFAULT ... ON CONVERSION ERROR is optional. If present, that result will be used if any errors are encountered in the string values. If omitted, then the error will be reported at runtime.

Duration Format

The International Organization for Standardization (ISO) defines a Date and time format – ISO 8601. This standard includes the representation of duration or periods. Oracle Rdb supports this format for use with the functions TO_YMININTERVAL and TO_DSINTERVAL.

A duration represents an interval between two date and time values. Oracle Rdb supports a subset of the ISO formatting that can be converted to the INTERVAL YEAR TO MONTH or the INTERVAL DAY TO SECOND data type.

```
[ + | - ] PnYnMnDTnHnMnS
```


The character string can begin with an optional "+" or "-" sign to indicate the sign of the interval. The next character must be an uppercase P, which indicates that the string is an ISO 8601 string. The capital letters P, Y, M, D, T, H, M, and S are format indicators for each of the date and time elements. Leading and trailing spaces are ignored, but embedded spaces will generate an error.

- ◆ P is the duration designator (for period) placed at the start of the duration representation.
- ◆ nY is the year designator that follows the value for the number of years.
- ◆ nM is the month designator that follows the value for the number of months.
- ◆ nD is the day designator that follows the value for the number of days.
- ◆ T is the time designator that precedes the time components of the representation. Note that T must be present only when there are time (H, M, S) elements in the string.
- ◆ nH is the hour designator that follows the value for the number of hours.
- ◆ nM is the minute designator that follows the value for the number of minutes.
- ◆ nS is the second designator that follows the value for the number of seconds.

If any of the value designators (Y, M, D, H, M, and S) are omitted, then the field they represent will be assumed zero (0). The n represents the unsigned value of the field and must only contain numeric digits (0 through 9). The exception is the seconds (S) field which represents seconds with decimal fractions that may include a decimal indicator (either "." or ","). At least one element of the interval must be represented; no element may be repeated.

Some examples:

- ◆ "P3Y6M4DT12H30M5.6S" represents a duration of three years, six months, four days, twelve hours, thirty minutes, and five point six seconds.
- ◆ "P1Y" represents a duration of one year.
- ◆ "PT50000S" represents a duration of fifty thousand seconds, which will be represented as thirteen hours fifty three minutes and twenty seconds.
- ◆ "P1MT1M" represents one month and one minute. The "T" separator is required to avoid ambiguity between months and minutes. This duration is accepted by Oracle Rdb as a year-month interval as input TO_YMINTERVAL but the low precision fields (D, H, M, S) will be discarded.

6.1.14 New CREATE DEFAULT AUDIT Statement

CREATE DEFAULT AUDIT Statement

This statement creates a system template object that is used to provide the audit and alarm characteristics for a newly created object of that type. For example, when a user creates a new SEQUENCE in the database, the audit/alarm characteristics are inherited automatically from the template without further action by the database administrator.

This statement can be used to create templates for:

- ◆ functions and procedures
- ◆ modules
- ◆ sequences
- ◆ tables and views

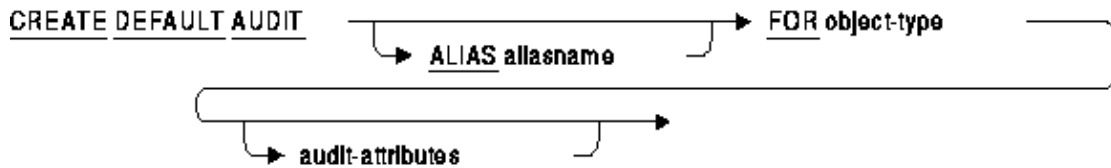
See also ALTER DEFAULT AUDIT Statement and the DROP DEFAULT AUDIT Statement.

Environment

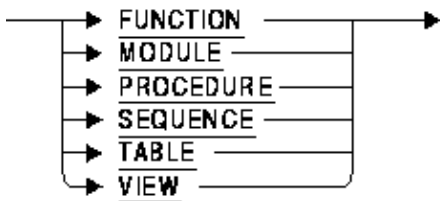
You can use the CREATE DEFAULT AUDIT statement:

- ◆ In interactive SQL
- ◆ Embedded in host language programs
- ◆ As part of a procedure in an SQL module
- ◆ In dynamic SQL as a statement to be dynamically executed

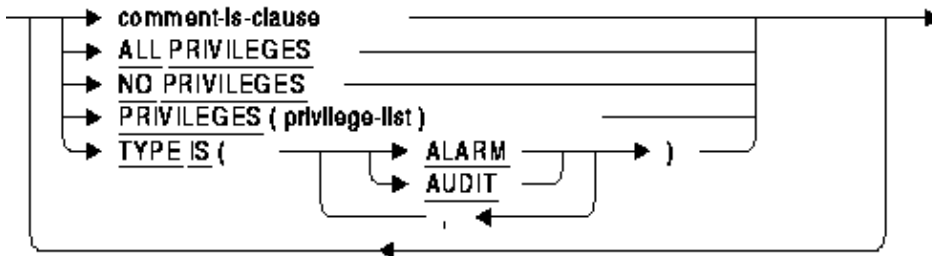
Syntax



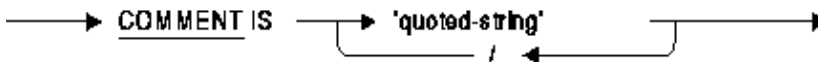
object-type =



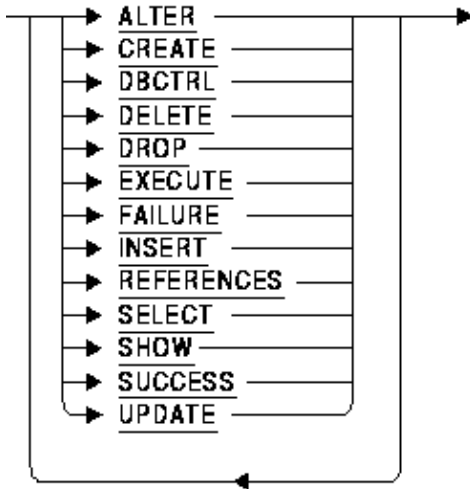
audit-attributes =



comment-is-clause =



privilege-list =



Arguments

- ◆ ALIAS aliasname
The name of the database alias if there is no default database for this session.
- ◆ COMMENT IS
Adds a comment to the template object.
- ◆ FOR object-type
The type of template.
- ◆ PRIVILEGES (privilege-list)
ALL PRIVILEGES
NO PRIVILEGES
Specifies the privileges that trigger audit or alarm actions.
The ALL PRIVILEGES clause is equivalent to specifying PRIVILEGES (ALL). NO PRIVILEGES is the default if none of these clauses is specified.
- ◆ TYPE IS (ALARM)
TYPE IS (AUDIT)
TYPE IS (ALARM, AUDIT)
Defines the scope for the privileges checking. The default if this clause is not specified is both ALARM and AUDIT.

Usage Notes

- ◆ You must have the SECURITY privilege on the database to execute the CREATE DEFAULT AUDIT statement.
- ◆ If the privilege set for AUDIT is different from ALARM then follow the CREATE DEFAULT AUDIT Statement with an ALTER DEFAULT AUDIT Statement. For example:

```
SQL> create default audit for function
cont> type is ( alarm )
cont> privilege ( drop );
SQL> alter default audit for function
```

```
cont> type is ( audit )
cont> privilege ( all );
```

- ◆ The clauses PRIVILEGES (ALL) and ALL PRIVILEGES are synonymous. The actual privileges applied to the audit will be filtered by those applicable to the object type. For example, EXECUTE privilege will not be applied to a table.
- ◆ The created templates are system owned and you must use the ALTER DEFAULT AUDIT Statement or DROP DEFAULT AUDIT Statement to manage them. The GRANT and REVOKE Statements can be used on the system objects but require appropriate DBCTRL privilege access.
- ◆ This statement will create system objects with the following names:
 - ◇ for sequence; RDB\$DEFAULT_AUDIT_SEQUENCE
 - ◇ for table; RDB\$DEFAULT_AUDIT_TABLE
 - ◇ for module; RDB\$DEFAULT_AUDIT_MODULE
 - ◇ for function; RDB\$DEFAULT_AUDIT_FUNCTION
 - ◇ for procedure; RDB\$DEFAULT_AUDIT_PROCEDURE
 - ◇ for view; RDB\$DEFAULT_AUDIT_VIEW
- ◆ Additionally, these template objects are used to provide the default access control list (ACL) for a new object. You can use the GRANT and REVOKE statements to manage the entries. This example grants privileges to a role (rights identifier) and a user:

```
SQL> create default audit for sequence
cont> ;
SQL>
SQL> grant select, show on sequence rdb$default_audit_sequence
cont>     to admin_user, user2
cont> ;
SQL>
SQL> show protection on sequence rdb$default_audit_sequence;
Protection on Sequence RDB$DEFAULT_AUDIT_SEQUENCE
  ( IDENTIFIER=[DEV,USER2],ACCESS=SELECT+SHOW)
  ( IDENTIFIER=ADMIN_USER,ACCESS=SELECT+SHOW)
  ( IDENTIFIER=[DEV,USER1],ACCESS=SELECT+SHOW+ALTER+DROP+DBCTRL+REFERENCES)
  ( IDENTIFIER=[*,*],ACCESS=NONE)
SQL>
```

The resulting access control list will implicitly grant the owner (USER1) all privileges and public no access. The added entries for ADMIN_USER and USER2 will be included for all new sequences.

In this example, a new sequence is created and inherits the new default access control list.

```
SQL> create sequence next_dept_id;
SQL>
SQL> show protection on sequence next_dept_id;
Protection on Sequence NEXT_DEPT_ID
  ( IDENTIFIER=[DEV,USER2],ACCESS=SELECT+SHOW)
  ( IDENTIFIER=ADMIN_USER,ACCESS=SELECT+SHOW)
  ( IDENTIFIER=[DEV,USER1],ACCESS=SELECT+SHOW+ALTER+DROP+DBCTRL+REFERENCES)
  ( IDENTIFIER=[*,*],ACCESS=NONE)
SQL>
```

- ◆ This statement creates fully functional database objects. However, Oracle recommends that these templates not be used as part of production views, triggers, etc. Oracle reserves the right to change the definitions of these objects in future releases of Oracle Rdb.

6.1.15 New ALTER DEFAULT AUDIT Statement

ALTER DEFAULT AUDIT Statement

This statement alters a system template object that is used to provide the audit and alarm characteristics for a newly created object of that type. For example, when a user creates a new SEQUENCE in the database, the audit/alarm characteristics are inherited automatically from the template without further action by the database administrator.

This statement can be used to alter templates for:

- ◆ functions and procedures
- ◆ modules
- ◆ sequences
- ◆ tables and views

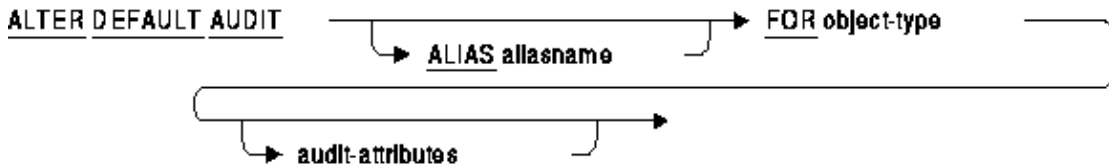
See also CREATE DEFAULT AUDIT Statement and the DROP DEFAULT AUDIT Statement.

Environment

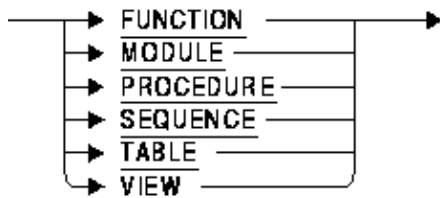
You can use the ALTER DEFAULT AUDIT statement:

- ◆ In interactive SQL
- ◆ Embedded in host language programs
- ◆ As part of a procedure in an SQL module
- ◆ In dynamic SQL as a statement to be dynamically executed

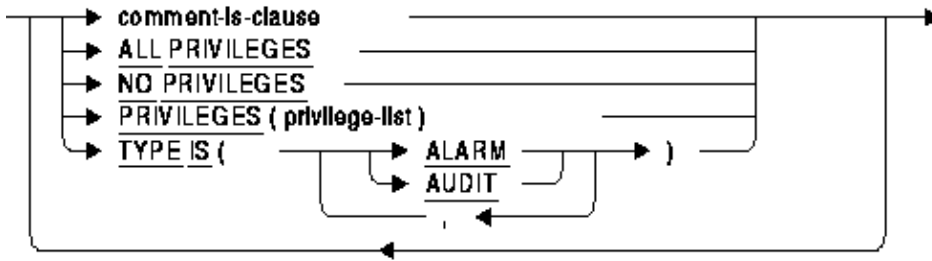
Syntax



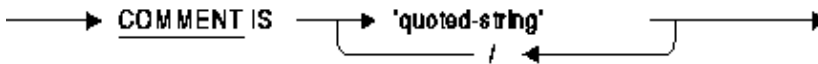
object-type =



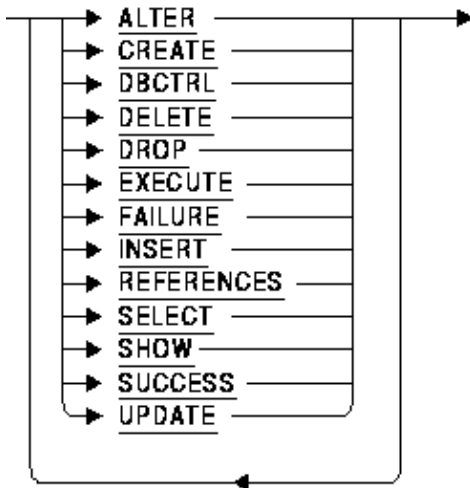
audit-attributes =



comment-is-clause =



privilege-list =



Arguments

- ◆ **ALIAS aliasname**
The name of the database alias if there is no default database for this session.
- ◆ **COMMENT IS**
Adds a comment to the template object.
- ◆ **FOR object-type**
The type of template.

- ◆ PRIVILEGES (privilege–list)
 ALL PRIVILEGES
 NO PRIVILEGES
 Specifies the privileges that trigger audit or alarm actions.
 The ALL PRIVILEGES clause is equivalent to specifying PRIVILEGES (ALL). NO PRIVILEGES is the default if none of these clauses is specified.
- ◆ TYPE IS (ALARM)
 TYPE IS (AUDIT)
 TYPE IS (ALARM, AUDIT)
 Defines the scope for the privileges checking. The default if this clause is not specified is both ALARM and AUDIT.

Usage Notes

- ◆ You must have the SECURITY privilege on the database to execute the ALTER DEFAULT AUDIT statement.
- ◆ The clauses PRIVILEGES (ALL) and ALL PRIVILEGES are synonymous. The actual privileges applied to the audit will be filtered by those applicable to the object type. For example, EXECUTE privilege will not be applied to a table.
- ◆ The default audit templates are system owned and you must use the ALTER DEFAULT AUDIT Statement or DROP DEFAULT AUDIT Statement to manage them. The GRANT and REVOKE Statements can be used on the system objects but require appropriate DBCTRL privilege access.

6.1.16 New DROP DEFAULT AUDIT Statement

DROP DEFAULT AUDIT Statement

This statement drops a system template object that is used to provide the audit and alarm characteristics for a newly created object of that type.

This statement can be used to drop templates for:

- ◆ functions and procedures
- ◆ modules
- ◆ sequences
- ◆ tables and views

See also CREATE DEFAULT AUDIT Statement and the ALTER DEFAULT AUDIT Statement.

Environment

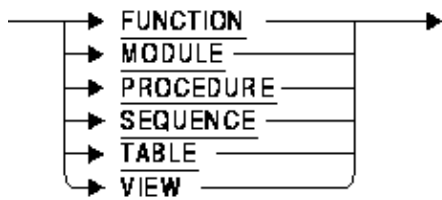
You can use the DROP DEFAULT AUDIT statement:

- ◆ In interactive SQL
- ◆ Embedded in host language programs
- ◆ As part of a procedure in an SQL module
- ◆ In dynamic SQL as a statement to be dynamically executed

Syntax



object-type =



Arguments

- ◆ ALIAS aliasname
 The name of the database alias if there is no default database for this session.

Usage Notes

- ◆ You must have the SECURITY privilege on the database to execute the DROP DEFAULT AUDIT statement.

6.1.17 New SESSION and GLOBAL Attributes for Sequences

This release of Oracle Rdb supports new keywords; SESSION and GLOBAL. These keywords are supported for use with CREATE SEQUENCE and ALTER SEQUENCE.

- ◆ SESSION
 Specify SESSION to create a session sequence, which is a special type of sequence that is specifically designed to be used with temporary tables that have session visibility. Unlike the existing regular sequences (referred to as "global" sequences), a session sequence returns a unique range of sequence numbers only within a session, but not across sessions. Another difference is that session sequences are not persistent. If a session goes away, so does the state of the session sequences that were accessed during the session.
 You may not specify both SESSION and GLOBAL.
- ◆ GLOBAL
 Specify GLOBAL to create a global, or regular, sequence. This is the default.
 You may not specify both GLOBAL and SESSION.

The following example shows the creation of a SESSION sequence.

```
SQL> create sequence Example
cont>     nomaxvalue
cont>     session
cont>     start with 456;
SQL>
SQL> show sequence Example;
```



```

EXAMPLE
Sequence Id: 4
Initial Value: 456
Minimum Value: 1
Maximum Value: (none)
Next Sequence Value: 456
Increment by: 1
Cache Size: 20
No Order
No Cycle
No Randomize
Session (local temporary sequence)
Wait

SQL>

```

6.1.18 New LogMiner Feature to Close and Immediately Reopen Table Output Files

The Oracle Rdb RMU Unload After_journal command can create large table output files when extracting data records from committed transactions recorded in Rdb database After Image Journal files. This new feature allows limiting the size of table output files so they can be processed more often by the database user without terminating the RMU Unload After_journal command data extraction operation. The current output file is closed and then "reopened" by creating a new output file with the same name but an incremented OpenVMS file version number, based on the specified table output file maximum record count integer value. The RMU Unload After_journal extraction operation will continue without interruption or loss of any data. The contents of the reopened output file will not contain any of the data records of the previous closed file but will continue from the point at which the previous file terminated.

The table output file will be closed and then reopened by default when the next transaction commit boundary is reached or, as an option, immediately when the specified maximum record count value is reached. The reopen record count is not an exact count of all records written to the table output file, which can vary based on the output file data format and other options that can be specified by the RMU Unload After_journal command. It is meant to be an approximate estimate made by the user of the desired size of each reopened output file and is the sum of the Delete and Modify data records written to the table output file. If the default option of ending the current output file at the next transaction commit boundary is in effect, Delete and Modify data records will continue to be output from the point at which the specified reopen record count is reached until the end of the current transaction.

The syntax for the new LogMiner feature to close and immediately reopen table output files is an option of the RMU Unload After_journal command TABLE qualifier which specifies the name of a table to be unloaded or multiple tables if wild card characters are included in the table name. The OUTPUT=file-spec option must also be specified as an option of the TABLE qualifier to name the table output file to be reopened. If the same output file name is specified for multiple tables, all of those tables will write Modify and Delete records to the same output file and all of these records will be included in the reopen count. The reopen syntax must be specified with the first TABLE qualifier to name the same output file or it will be ignored. The INCLUDE=ACTION=(NODELETE,NOMODIFY) qualifier syntax cannot also be included in the same RMU Unload After_journal command since this would exclude both the Modify and Delete data

records from the output file which are required to determine when the output file can be reopened.

The command line syntax of the LogMiner option to close and immediately reopen table output files specified as an option of the /TABLE qualifier is:

```
REOPEN=(RECORDS=integer,[NO]COMMIT_BOUNDARY)
```

"RECORDS=integer" specifies an unsigned integer value between 1 and 2147483647 which is the number of Delete and Modify data records that the Rdb RMU Unload After_journal command can write to a table output file before the output file is closed and immediately reopened for output by creating a new output file with the same name but an incremented OpenVMS file version number. The reopen will happen by default when the next transaction commit boundary is reached or, as an option, immediately when the specified maximum record count value is reached.

"COMMIT_BOUNDARY" is the default which specifies that the file is closed and reopened at the end of the current transaction. If "NOCOMMIT_BOUNDARY" is specified, the file will immediately be closed and then reopen when the reopen count is reached. The RMU Unload After_journal command will continue to execute without interruption during the reopen process.

The reopen parameters can also be specified in the table's option file designated by the "/OPTIONS=file_spec" command line qualifier of the RMU Unload After_journal command.

```
TABLE=table_name,REOPEN_RECORDS=integer,REOPEN_COMMIT=TRUE | FALSE
```

"REOPEN_COMMIT=TRUE" specifies the default option to close and reopen the table output file when the current transaction is committed. If "REOPEN_COMMIT=FALSE" is specified, the table output file is immediately closed when the specified reopen count is reached.

The following example shows a table options file with each table writing to a different output file. The first table closes and reopens the output file as soon as the reopen record count is reached. The second table closes and reopens the output file when the current transaction is committed which is the default action.

```
$ type LOGMINER_TABLES.OPT
table=invoices,output=reports.dat,reopen_records=5,reopen_commit=false
table=costs,output=cost_analysis.dat,reopen_records=5,reopen_commit=true
```

In the following example, the data records for two database tables are written to the same output file. Only the reopen parameters specified for the first table designated which writes to the same output are used so reopen parameters do not need to be specified for any other tables writing to the same file. The reopen Modify and Delete data record count specified is 5 and the close and reopen of the table output file occurs as soon as the reopen record count is reached. The log messages show that 9 output files, REPORTS.DAT;1 to REPORTS.DAT;9, are created during the extraction operation, which continues without interruption. The %RMU-I-LMREOPENCOUNTS informational log message names the table "INVOICES", which is the first table specified as writing to the REPORTS.DAT table output file, but the number of Modify and Delete data records written in the message includes any other tables writing to the named file (in this case the "COSTS" table). The messages put out when the RMU Unload After_journal command terminates specify the total number of Modify and Delete data records written to all the created REPORTS.DAT files by each individual database table.

```
$ rmu/unload/after_journal/statistics/FORMAT=DUMP/TRACE/LOG -
   accounting.rdb accounting_jrnl.aij -
   /table=(name=invoices, -
```

Oracle® Rdb for OpenVMS

```
output=reports.dat, -
REOPEN=(RECORDS=5,NOCOMMIT_BOUNDARY), -
table_definition=invoices_def, -
record_definition=rdb_lm_invoices ) -
/table=(name=costs, -
record_definition=rdb_lm_costs, -
table_definition=costs_def, -
output=reports.dat)
%RMU-I-UNLAIJFL, Unloading table INVOICES to
DEVICE:[DIRECTORY]REPORTS.DAT;1
%RMU-I-UNLAIJFL, Unloading table COSTS to
DEVICE:[DIRECTORY]REPORTS.DAT;1
%RMU-I-LOGOPNAIJ, opened journal file
DEVICE:[DIRECTORY]ACCOUNTING_JRNL.AIJ;1 at 29-JAN-2019 13:55:53.06
%RMU-I-AIJRSTSEQ, journal sequence number is "0"
29-JAN-2019 13:55:53.06 Starting at offline open record sequence number 0
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 256 committed
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 257 committed
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;1" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 5 data records written,
specified reopen count is 5
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;2" created
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 258 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 259 committed
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;2" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 5 data records written,
specified reopen count is 5
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;3" created
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 261 committed
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 262 committed
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 263 committed
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;3" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 5 data records written,
specified reopen count is 5
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 264 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;4" created
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 266 committed
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;4" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 5 data records written,
specified reopen count is 5
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 267 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;5" created
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 268 committed
```

Oracle® Rdb for OpenVMS

```
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 269 committed
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;5" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 5 data records written,
specified reopen count is 5
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;6" created
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 270 committed
%RMU-I-LOGRECSTAT, transaction with TSN 271 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 272 committed
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;6" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 5 data records written,
specified reopen count is 5
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 260 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;7" created
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 274 committed
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 275 committed
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;7" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 5 data records written,
specified reopen count is 5
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;8" created
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 276 committed
%RMU-I-LOGRECSTAT, transaction with TSN 265 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 278 committed
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;8" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 5 data records written,
specified reopen count is 5
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]INVOICES.DAT;9" created
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 279 committed
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 273 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 281 committed
%RMU-I-LOGRECSTAT, transaction with TSN 283 committed
%RMU-I-LOGRECSTAT, transaction with TSN 282 committed
%RMU-I-LOGRECSTAT, transaction with TSN 286 committed
%RMU-I-LOGRECSTAT, transaction with TSN 288 committed
%RMU-I-EXTSRTSTAT, Records:7 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 289 committed
%RMU-I-LOGRECSTAT, transaction with TSN 291 committed
%RMU-I-AIJMODSEQ, next AIJ file sequence number will be 1
%RMU-I-LOGSUMMARY, total 30 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
-----
ELAPSED:      0 00:00:00.09 CPU: 0:00:00.01 BUFIO: 167 DIRIO: 105 FAULTS: 186
Table "INVOICES" : 35 records written (33 modify, 2 delete)
Table "COSTS" : 9 records written (9 modify, 0 delete)
```

Oracle® Rdb for OpenVMS

Total : 44 records written (42 modify, 2 delete)
\$

In the following example, the data records for two database tables are written to the same output file. Only the reopen parameters specified for the first table, which writes to the same output file, are used so reopen parameters do not need to be specified for any other tables writing to the same file. The reopen Modify and Delete data record count specified is 5 and the close and reopen of the table output file occurs at the end of the current transaction, the default, not as soon as the reopen record count is reached. The log messages show that 8 output files, REPORTS.DAT;1 to REPORTS.DAT;8, are created and that in some cases more than 5 Modify and Delete data records are written because the current transaction does not end immediately after the reopen record count is reached. The %RMU-I-LMREOPENCOUNTS informational log message names the table "INVOICES", which is the first table specified as writing to the REPORTS.DAT table output file, but the number of Modify and Delete data records written in the message includes any other tables writing to the named file (in this case the "COSTS" table). The messages put out when the RMU Unload After_journal command terminates specify the total number of Modify and Delete data records written to all the created REPORTS.DAT files by each individual database table.

```
$ rmu/unload/after_journal/statistics/FORMAT=DUMP/TRACE/LOG -
    accounting.rdb accounting_jrnl.aij -
    /table=(name=invoices, -
    output=reports.dat, -
    REOPEN=(RECORDS=5), -
    table_definition=invoices_def, -
    record_definition=rdb_lm_invoices ) -
    /table=(name=costs, -
    record_definition=rdb_lm_costs, -
    table_definition=costs_def, -
    output=reports.dat)
%RMU-I-UNLAIJFL, Unloading table INVOICES to DEVICE:[DIRECTORY]REPORTS.DAT;1
%RMU-I-UNLAIJFL, Unloading table COSTS to DEVICE:[DIRECTORY]REPORTS.DAT;1
%RMU-I-LOGOPNAIJ, opened journal file
    DEVICE:[DIRECTORY]ACCOUNTING_JRNL.AIJ;1 at 29-JAN-2019 13:45:07.66
%RMU-I-AIJRSTSEQ, journal sequence number is "0"
29-JAN-2019 13:45:07.66 Starting at offline open record sequence number 0
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 256 committed
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 257 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
    "DEVICE:[DIRECTORY]REPORTS.DAT;1" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 6 data records written,
    specified reopen count is 5
%RMU-I-LOGRECSTAT, transaction with TSN 258 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
    "DEVICE:[DIRECTORY]REPORTS.DAT;2" created
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 259 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
    "DEVICE:[DIRECTORY]REPORTS.DAT;2" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 6 data records written,
    specified reopen count is 5
%RMU-I-LOGRECSTAT, transaction with TSN 261 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
    "DEVICE:[DIRECTORY]REPORTS.DAT;3" created
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 262 committed
```

Oracle® Rdb for OpenVMS

```
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 263 committed
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 264 committed
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;3" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 5 data records written,
specified reopen count is 5
%RMU-I-LOGRECSTAT, transaction with TSN 266 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;4" created
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 267 committed
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 268 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;4" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 7 data records written,
specified reopen count is 5
%RMU-I-LOGRECSTAT, transaction with TSN 269 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;5" created
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 270 committed
%RMU-I-LOGRECSTAT, transaction with TSN 271 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 272 committed
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LMREOPENCLOSE, Reopen record count reached,
output file "DEVICE:[DIRECTORY]REPORTS.DAT;5" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 6 data records written,
specified reopen count is 5
%RMU-I-LOGRECSTAT, transaction with TSN 260 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;6" created
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 274 committed
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 275 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;6" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 6 data records written,
specified reopen count is 5
%RMU-I-LOGRECSTAT, transaction with TSN 276 committed
%RMU-I-LOGRECSTAT, transaction with TSN 265 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached,
output file "DEVICE:[DIRECTORY]REPORTS.DAT;7" created
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 278 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LMREOPENCLOSE, Reopen record count reached,
output file "DEVICE:[DIRECTORY]REPORTS.DAT;7" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 5 data records written,
specified is 5
%RMU-I-LOGRECSTAT, transaction with TSN 279 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
"DEVICE:[DIRECTORY]REPORTS.DAT;8" created
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 273 committed
```

Oracle® Rdb for OpenVMS

```
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 281 committed
%RMU-I-LOGRECSTAT, transaction with TSN 283 committed
%RMU-I-LOGRECSTAT, transaction with TSN 282 committed
%RMU-I-LOGRECSTAT, transaction with TSN 286 committed
%RMU-I-LOGRECSTAT, transaction with TSN 288 committed
%RMU-I-EXTSRTSTAT, Records:7 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 289 committed
%RMU-I-LOGRECSTAT, transaction with TSN 291 committed
%RMU-I-AIJMODSEQ, next AIJ file sequence number will be 1
%RMU-I-LOGSUMMARY, total 30 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
-----
ELAPSED:      0 00:00:00.18 CPU: 0:00:00.06 BUFIO: 155 DIRIO: 91 FAULTS: 187
Table "INVOICES" : 35 records written (33 modify, 2 delete)
Table "COSTS" : 9 records written (9 modify, 0 delete)
Total : 44 records written (42 modify, 2 delete)
$
```

Chapter 7

Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.0

7.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.0

7.1.1 Relaxed Type Checking for DEFAULT Clause

In prior versions of Oracle Rdb, the DEFAULT for a domain or column was strictly checked when the type was DATE, TIME, TIMESTAMP or INTERVAL. With this release, these rules for DEFAULT have been relaxed and allow compatible types to be used.

- ◆ TIMESTAMP can now have a DEFAULT with the data types DATE (ANSI), DATE (VMS) and TIMESTAMP
- ◆ DATE VMS can now have a DEFAULT with the data types DATE (ANSI), DATE (VMS) and TIMESTAMP
- ◆ INTERVAL can now have a DEFAULT with the same interval qualifier or a subset of the columns interval qualifier. For example, if the column is defined as INTERVAL YEAR TO MONTH then the DEFAULT can be an expression that results in INTERVAL YEAR, or INTERVAL MONTH.

The following example shows ALTER TABLE statements that failed in prior versions but which are now accepted by SQL.

```
SQL> set dialect 'sql99';
SQL>
SQL>
SQL> create table INFO
cont>     (seq_no integer identity
cont>     );
SQL>
SQL> alter table INFO
cont>     add column dt timestamp default current_date
cont> ;
%SQL-F-DEFVALINC, You specified a default value for DT which is inconsistent
with its data type
SQL>
SQL> alter table INFO
cont>     add column duration interval day to second(2) default interval'0'day
cont> ;
%SQL-F-DEFVALINC, You specified a default value for DURATION which is
inconsistent with its data type
SQL>
```

7.1.2 New Statistics Screen Shows Top Processes Accessing a Table Logical Area

Bug 18460947

A new Oracle Rdb RMU/SHOW STATISTICS Zoom screen option has been added to the existing Logical Area Information Logical Area Statistics screen to display the top ten or fewer processes attached to an Oracle Rdb database that are accessing the table logical area currently displayed in the Logical Area Statistics screen header. When the new Zoom option, displayed at the bottom of the

Oracle® Rdb for OpenVMS

Logical Area Statistics screen, is selected a menu will appear on the left side of the screen to allow one of the statistics listed in the menu to be chosen to select the top processes accessing the table logical area. The last choice in the menu will be "ALL", indicating that the sum of all the individual statistics listed above it in the menu will be used to select the top processes accessing the table logical area.

When the menu statistics entry is selected, a maximum of ten process IDs will be displayed in a Zoom screen sorted in descending order based on the current value of the chosen statistic. The process ID will end with a colon followed by the stream ID assigned by the database and the letter "A" indicating that the process has been activated and attached for global statistics collection. The numerical statistic value used to select the process will be displayed next to each process ID.

This feature is only available for table logical areas. As with all screens that display per-process statistics, process statistic collection must be enabled for the database.

The following example shows the RMU/SHOW STATISTICS screen displays that implement this new functionality. The right sides and blank portions of these screens are not shown to save space. The first screen shown is the Logical Area Statistics screen for the EMPLOYEES table EMPIDS_LOW logical area in the MF_PERSONNEL database with the new Zoom screen "Zoom" option at the bottom of the screen. The second screen shows the menu that will appear on the left of the Logical Area Statistics screen when the user selects the new "Zoom" option. This menu allows the user to select the statistic to be used to determine the top ten or fewer processes accessing the table logical area based on the selected statistic name. The last "ALL" entry on the menu will use the sum of all the statistics listed above it to select the top processes accessing the table. Once the menu choice has been selected, in this case "record fetched", the third Zoom screen will be displayed which shows the process ID and process statistic value sorted in descending order of the process statistic value.

```

$ RMU/SHOW STATISTICS MF_PERSONNEL

Node: TSTNOD (1/1/16)                               Oracle Rdb V7.3-300 Perf. Monitor
Rate: 3.00 Seconds                                  Logical Area Statistics
Page: 1 of 1                                         DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
.....
Table EMPLOYEES in EMPIDS_LOW

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

record marked                0          0          0.0          0          0.0
record fetched                1          0          0.6          222         74.0
  fragmented                  0          0          0.0          0          0.0
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                 0          0          0.0          0          0.0
  fragmented                  0          0          0.0          0          0.0
sequential scan               0          0          0.0          0          0.0
record fetched                0          0          0.0          0          0.0

.....
Config Exit Graph Help Menu Options Pause Reset Set_rate Time_plot Write X_plot
Zoom !

```

Oracle® Rdb for OpenVMS

Node: TSTNOD (1/1/16)
 Rate: 3.00 Seconds
 Page: 1 of 1

Oracle Rdb V7.3-300 Perf. Monitor
 Logical Area Statistics
 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1

Table EMPLOYEES in EMPIDS_LOW

statistic..... name.....	rate.per.second.....		total.....	average.....
	max.....	cur.....		
A. record marked	0	0	0.0	0
B. record fetched	1	0	0.5	222
C. fragmented	0	0	0.0	0
D. record stored	0	0	0.0	0
E. fragmented	0	0	0.0	0
F. pages checked	0	0	0.0	0
G. saved IO	0	0	0.0	0
H. discarded	0	0	0.0	0
I. record erased	0	0	0.0	0
J. fragmented	0	0	0.0	0
K. sequential scan	0	0	0.0	0
L. record fetched	0	0	0.0	0
M. ALL				

.....
 Type <return> or <letter> to select logical area statistics, <control-Z> to cancel

Node: TSTNOD (1/1/16)
 Rate: 3.00 Seconds
 Page: 1 of 1

Oracle Rdb V7.3-300 Perf. Monitor
 Logical Area Statistics
 DEVICE:[DIRECORY]MF_PERSONNEL.RDB;1

Table EMPLOYEES in EMPIDS_LOW

...Top Processes Accessing Logical Area EMPLOYEES in EMPIDS_LOW.....

top.....	statistic
processes.....	record fetched
. 20B54CFC:1A	12036
. 20B3EB03:1A	10347
. 20ADC254:1A	10321
. 20A8CE4A:1A	9560
. 20B31240:1A	8374
. 20B5743A:1A	7312
. 20B39E2F:1A	6543
. 20A71A11:1A	5478
. 20AFC603:1A	4312
. 20B46D41:1A	3245

.....
 Type any key to erase display and return to logical area statistics menu

7.1.3 Relaxed Naming Rules for RMU Extract Option=MATCH Option

In prior releases of Oracle Rdb, the RMU Extract Option=MATCH option required that names include a trailing "%" wildcard in order to match a single object name in the database.

The following example shows the problem if the wildcard is missing.

```
$ rmu/extract-
  /item=table-
  /option=(noheader,filename_only,match:work_status) -
  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 PERSONNEL';
-- no tables defined
$
```

This problem has been corrected in Oracle Rdb Release 7.3.3.0. In this release, RMU recognizes that this is a fixed length name and adds trailing spaces to enable the single object match. The following example shows the simplified interface.

```
$ rmu/extract-
  /item=table-
  /option=(noheader,filename_only,match:work_status) -
  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 PERSONNEL';
create table WORK_STATUS (
  STATUS_CODE STATUS_CODE,
  STATUS_NAME STATUS_NAME,
  STATUS_TYPE STATUS_TYPE);

commit work;
$
```

7.1.4 RMU/RESTORE Now Always Displays the %RMU-I-AIJRECFUL Message

Bug 14375975

With this release of Oracle Rdb, the RMU message RMU-I-AIJRECFUL is always displayed by RMU Restore when after image journaling is enabled for the database. Previously, this message was only output if the RMU Restore /LOG qualifier was specified.

The RMU Recover command is often executed after an RMU Restore operation to apply the contents of one or more after image journal (AIJ) files to update the database with any changes made since the database backup file was created. RMU Restore displays, using the AIJRECFUL message, the AIJ file sequence number of the first AIJ file where the recovery should start. This is important information for the database administrator to determine where a database recovery should start and to

make sure that AIJ files are applied to the database in the correct order.

Note

The RMU/Dump/After_journal/ONLY=TYPE=OPEN command can be used to dump the AIJ sequence number contained in the Open records of AIJ files.

Examples

In the following example, the database ABC, previously backed up by the RMU/BACKUP command with circular after image journaling enabled, is restored by the RMU/RESTORE/NOLOG command.

The RMU-I-AIJREFCUL message states that the next recovery of the database should start with the AIJ file which has the sequence number "0" specified in its Open record. An RMU/DUMP/AFTER_JOURNAL dump of the AIJABC1.AIJ file shows that this AIJ file has a "0" sequence number in its Open record and belongs to the ABC database. The RMU/RECOVER command is then used to bring the database up to date by applying journaled changes made to the database since the database backup file was created (contained in the AIJABC1.AIJ after image journal file).

```

$ RMU/RESTORE/NOLOG/NORECOVER/NOLOG ABC_SAVE.RBF
%RMU-I-AIJREFCUL, Recovery of the entire database starts with AIJ file
sequence 0
%RMU-I-AIJRSTAVL, 3 after-image journals available for use
%RMU-I-AIJRSTMOD, 1 after-image journal marked as "modified"
%RMU-I-AIJISON, after-image journaling has been enabled
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
$!
$ RMU/DUMP/AFTER/ONLY=TYPE=OPEN AIJABC1.AIJ
-----
* Oracle Rdb V7.3-220                                17-JUL-2017 15:27:49.16
*
* Dump of After Image Journal
*   Filename: DEVICE:[DIRECTORY]AIJABC1.AIJ;1
*
*-----
1/1          TYPE=0, LENGTH=510, TAD=17-JUL-2017 15:27:48.47, CSM=00
Database DEVICE:[DIRECTORY]ABC.RDB;1
Database timestamp is 17-JUL-2017 15:27:47.03
Facility is "RDMSAIJ ", Version is 721.0
Database version is 73.0
AIJ Sequence Number is 0
Last Commit TSN is 96
Synchronization TSN is 0
Journal created on VMS platform
Type is Normal (unoptimized)
Open mode is Initial
Backup type is Active
I/O format is Record
Commit-to-Journal optimization disabled
AIJ journal activation ID is 00B1E299B4FE8A62
LogMiner is disabled

$!
$ RMU/RECOVER/NOLOG AIJABC1.AIJ
%RMU-I-LOGRECDB, recovering database file DEVICE:[DIRECTORY]ABC.RDB;1

```

```
%RMU-I-LOGRECSTAT, transaction with TSN 128 committed
%RMU-I-LOGRECSTAT, transaction with TSN 129 committed
%RMU-I-LOGRECSTAT, transaction with TSN 130 committed
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-AIJAUTOREC, starting automatic after-image journal recovery
%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed
%RMU-W-NOTRANAPP, no transactions in this journal were applied
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-AIJVNOSYNC, AIJ file DEVICE:[DIRECTORY]AIJABC1.AIJ;1 synchronized with
database
%RMU-I-AIJSUCCEC, database recovery completed successfully
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence number
needed will be 2
```

7.1.5 New SQL Built-in Functions

This release of Oracle Rdb introduces the following new built-in functions:

```
LTRIM
RTRIM
BTRIM
FIRST_VALUE
LAST_VALUE
LISTAGG
GROUP_CONCAT
```

7.1.5.1 New String Functions

In prior releases of Oracle Rdb, the SQL_FUNCTIONS script could be used to add a version of the RTRIM and LTRIM functions to an Rdb database. These definitions were limited in the character set they supported and always resulted in a VARCHAR (2000) result.

In this release of Rdb, new native versions of these two functions are now available. They have the advantage of being more efficient, accept any database character set and result in a VARCHAR string that is limited to the length of the source string.

- ◆ LTRIM (source_string [, trim_pattern])

This function trims the leading characters (left end) from the source_string that also appear in the trim_pattern. The trim_pattern defaults to a single space character from the source_string character set.

```
SQL> select ltrim (product_desc, '<>* ')
cont> from sample
cont> where product_desc containing 'Not';

Not Available **
Not Available >>
Not Available >>
3 rows selected
SQL>
```

Note

A similar LTRIM function is provided in the SQL_FUNCTIONS library and is now superseded by this built-in function. Applications compiled with SQL Precompiler or SQL Module Language will need to be recompiled to make use of this new function.

◆ RTRIM (source_string [, trim_pattern])

This function trims the trailing characters (right end) from the source_string that also appear in the trim_pattern. The trim_pattern defaults to a single space character from the source_string character set.

```
SQL> select rtrim (product_desc, '<>* ')
cont> from sample
cont> where product_desc containing 'Not';

** Not Available
<< Not Available
>> Not Available
3 rows selected
SQL>
```

Note

A similar RTRIM function is provided in the SQL_FUNCTIONS library and is now superseded by this built-in function. Applications compiled with SQL Precompiler or SQL Module Language will need to be recompiled to make use of this new function.

◆ BTRIM (source_string [, trim_pattern])

This function trims the trailing and leading characters (both ends) from the source_string that also appear in the trim_pattern. The trim_pattern defaults to a single space character from the source_string character set.

```
SQL> select btrim (product_desc, '<>* ')
cont> from sample
cont> where product_desc containing 'Not';

Not Available
Not Available
Not Available
3 rows selected
SQL>
```

Usage Notes

- ◆ These functions are similar to the TRIM function except that the trim_pattern can be longer than one character and thus a variety of characters can be trimmed from the string.
- ◆ If the value expression passed as source_string is not CHAR, VARCHAR, BINARY or VARBINARY, then Rdb will implicitly convert that value to VARCHAR before applying the trim action (as in the following example). *RTRIM (10000.000, '0')*
- ◆ If the query is executed under an ORACLE dialect and the result of the trim function is a zero length string, then this is assumed to be equivalent to a NULL result.
- ◆ If LTRIM and RTRIM are nested with the same trim_pattern, Rdb will attempt to substitute a call to the routine BTRIM that trims leading and trailing characters from the source_string (as in the following example). *LTRIM (RTRIM (last_name, ''), '')*

7.1.5.2 New Aggregate Functions

Aggregate functions can be used in the context of a GROUP BY clause, or can operate on the whole result table. Aggregate functions operate on non-NULL values of the source value-expression and they can also be modified with the FILTER (WHERE ...) clause.

FIRST_VALUE Function

This function returns the first value of the specified column or value-expression computed from the values of the rows in the group. The set of values within the group can be reordered using the WITHIN GROUP (ORDER BY ...) clause.

This function returns a result that matches the data type of the source expression. If, after applying the FILTER (WHERE ...) clause or the WHERE clause, there are no rows then the result will be NULL.

The WITHIN GROUP (ORDER BY ...) clause may be omitted but the results of the function are then not deterministic.

This example shows that the first value for SUPERVISOR_ID is determined after ordering by the EMPLOYEES job starting date (JOB_START).

Example 7-1 Using the FIRST_VALUE Function

```
SQL> select
cont>     EMPLOYEE_ID
cont>     ,FIRST_VALUE (SUPERVISOR_ID)
cont>         within group (order by JOB_START) as FIRST_BOSS
cont>     ,MIN (SUPERVISOR_ID)
cont> from JOB_HISTORY
cont> where EMPLOYEE_ID = '00167'
cont> group by EMPLOYEE_ID
cont> ;
  EMPLOYEE_ID  FIRST_BOSS
  00167        00248        00164
1 row selected
SQL>
```

The output of the MIN function is shown to demonstrate that the results might be different for the FIRST_VALUE function because of the ordering performed by the WITHIN GROUP (ORDER BY ...) clause.

LAST_VALUE Function

This function returns the last value of the specified column or value-expression computed from the values of the rows in the group. The set of values within the group can be reordered using the WITHIN GROUP (ORDER BY ...) clause.

This function returns a result that matches the data type of the source expression. If, after applying the FILTER (WHERE ...) clause or the WHERE clause, there are no rows then the result will be NULL.

The WITHIN GROUP (ORDER BY ...) clause may be omitted but the results of the function are then not deterministic.

This example uses `LAST_VALUE` to determine the `SUPERVISOR_ID` of the current job, which is selected using the `FILTER` clause.

Example 7–2 Using the `LAST_VALUE` Function

```
SQL> select
cont>     EMPLOYEE_ID
cont>     ,LAST_VALUE (SUPERVISOR_ID)
cont>       within group (order by JOB_START)
cont>       filter (where JOB_END is NULL)
cont>       as CURRENT_BOSS
cont> from JOB_HISTORY
cont> where EMPLOYEE_ID = '00167'
cont> group by EMPLOYEE_ID
cont> ;
EMPLOYEE_ID  CURRENT_BOSS
00167        00164
1 row selected
SQL>
```

LISTAGG Function

The `GROUP BY` clause creates a set of rows that match the grouping criteria. The values of a column, or value-expression computed from the values of a row in the group, may be concatenated forming a single string result. `LISTAGG` provides clauses to control its action in case the result is too long, as well as specification of the separator character string.

This function returns a `VARCHAR` result. Any column or value expression will be implicitly converted to `VARCHAR` prior to executing the `LISTAGG` function. If, after applying the `FILTER` (`WHERE ...`) clause or the `WHERE` clause, there are no rows, then the result will be `NULL`.

By default, `LISTAGG` returns a `VARCHAR(4000)` result, but this can be changed to a smaller or larger result using these options:

1. Interactive SQL and Dynamic SQL can use the `SET RESULT LENGTH` statement.
2. Specify `PRAGMA=RESULT_LENGTH:n` in the `/SQLOPTIONS` qualifier for the SQL Pre-compiler.
3. Specify `/PRAGMA=RESULT_LENGTH:n` qualifier for the SQL Module Language compiler.
4. Specify the `PRAGMA (RESULT LENGTH n)` clause in the `DECLARE MODULE` statement for either SQL Pre-compiler or SQL Module Language compiler context file.
5. Include the `PRAGMA (RESULT LENGTH n)` in the `MODULE` language header.

The value for `RESULT LENGTH` can range from 256 through to 32767.

The default `SEPARATOR` (when omitted) is an empty string. If the result exceeds the allocated buffer size, then an error will be raised that is the default `ON OVERFLOW ERROR`. The clause `ON OVERFLOW TRUNCATE` defaults to a truncation indicator of `'...'` and `WITH COUNT`. To eliminate the truncation indicator string, specify either an empty string (`''`) or `NULL`.

The `WITHIN GROUP (ORDER BY ...)` clause may be omitted but the results of the function are then not deterministic.

Note

Applications that use LISTAGG from a module written in C or C++ will have a symbol defined, SQL_PRAGMA_RESULT_LENGTH, that reflects the default value of RESULT_LENGTH, or the value defined by the SQLOPTIONS PRAGMA=RESULT_LENGTH option, or the setting in the DECLARE MODULE Statement within the PRAGMA clause. This symbol can be used to allocate memory to receive the result of the LISTAGG functions.

```

      .
      .
      .
long  SQLCODE;
char * lagg_result;

      lagg_result = malloc (SQL_PRAGMA_RESULT_LENGTH);

      .
      .
      .

exec sql
      select listagg (first_name, ';' )
              within group (order by middle_initial)
      into :lagg_result
      from employees
      where last_name = 'Smith'
      group by last_name
      ;
if (SQLCODE != 0) sql_signal();
      .
      .
      .

```

This example shows the list of employees with the LAST_NAME of 'Smith', and returns the FIRST_NAME of all employees that share the last name.

Example 7-3 Using the LISTAGG Function

```

SQL> select
cont>     LAST_NAME
cont>     ,LISTAGG (FIRST_NAME, '/' ON OVERFLOW TRUNCATE ' ' WITH COUNT)
cont>     WITHIN GROUP (order by FIRST_NAME, MIDDLE_INITIAL)
cont> from
cont>     EMPLOYEES
cont> where
cont>     LAST_NAME starting with 'Smith'
cont> group by
cont>     LAST_NAME
cont> ;
LAST_NAME
Smith      Roger      /Terry
1 row selected
SQL>

```

GROUP_CONCAT Function

The GROUP_CONCAT provides an alternate syntax for the LISTAGG functionality. It is provided

for compatibility with other SQL implementations. SQL transforms this function to a LISTAGG equivalent.

This function returns a VARCHAR result. All column and value expressions will be implicitly converted to VARCHAR prior to executing the GROUP_CONCAT function. GROUP_CONCAT allows a list of values to be passed. SQL implicitly generates a CONCAT function with these arguments and NULL values will be omitted. If, after applying the FILTER (WHERE ...) clause or the WHERE clause, there are no rows, then the result will be NULL.

By default, GROUP_CONCAT returns a VARCHAR(4000) result but this can be changed to a smaller or larger result using the options listed under LISTAGG function.

The default SEPARATOR is a ',' (comma). If the result exceeds the allocated buffer size, then it will be truncated (equivalent to the LISTAGG clause ON OVERFLOW TRUNCATE NO COUNT).

The ORDER BY clause within GROUP_CONCAT may be omitted but the results of the function are then not deterministic.

Note

Applications that use GROUP_CONCAT from a module written in C or C++ will have a symbol defined, SQL_PRAGMA_RESULT_LENGTH, that reflects the default value of RESULT_LENGTH, or the value defined by the SQLOPTIONS PRAGMA=RESULT_LENGTH option, or the setting in the DECLARE MODULE Statement within the PRAGMA clause. This symbol can be used to allocate memory to receive the result of the GROUP_CONCAT functions.

This example shows the list of employees with the LAST_NAME of 'Smith', and returns the FIRST_NAME of all employees that share the last name.

Example 7-4 Using the GROUP_CONCAT Function

```
SQL> select
cont>     LAST_NAME
cont>     ,GROUP_CONCAT (FIRST_NAME
cont>                     order by FIRST_NAME, MIDDLE_INITIAL
cont>                     SEPARATOR '/')
cont> from
cont>     EMPLOYEES
cont> where
cont>     LAST_NAME starting with 'Smith'
cont> group by
cont>     LAST_NAME
cont> ;
LAST_NAME
Smith      Roger      /Terry
1 row selected
SQL>
```

7.1.6 –RMU–F–DBROOTFILE, –RMU–F–DBDATAFILE messages output with %RMU–F–BADAIJFILE

It is a common user error for an Oracle Rdb database root file, area data file, or area data snapshot file to be specified instead of a database after image journal (AIJ) file when executing the RMU/RECOVER command.

Now a new –RMU–F–DBROOTFILE error message will be output as a secondary message when the existing %RMU–F–BADAIJFILE fatal error message is output if a database root file (*.RDB) is specified in an RMU/RECOVER command where a database after image journal file should be specified.

```
%RMU-F-BADAIJFILE, this file is not a valid after image journal file,
please correct the file specification
-RMU-F-DBROOTFILE, specify a database after image journal file, this
is a database root file
%RMU-F-FTL_RCV, Fatal error for RECOVER operation at 25-JUL-2017
10:02:46.39
```

If a database area data file (*.RDA) or area data snapshot file (*.SNP) is specified in an RMU/RECOVER command where a database after image journal file should be specified, a new –RMU–F–DBDATAFILE error message will be output as a secondary message when the existing %RMU–F–BADAIJFILE fatal error message is output.

```
%RMU-F-BADAIJFILE, this file is not a valid after image journal file,
please correct the file specification
-RMU-F-DBDATAFILE, specify a database after image journal file, this
is a database data file
%RMU-F-FTL_RCV, Fatal error for RECOVER operation at 25-JUL-2017
10:02:53.28
```

In the following example, the secondary "–RMU–F–DBROOTFILE" message is output following the primary "%RMU–F–BADAIJFILE" fatal error message if a database root file, MF_PERSONNEL.RDB, is specified instead of a database after image journal file in an RMU/RECOVER command, and the secondary "–RMU–F–DBDATAFILE" message is output following the primary "%RMU–F–BADAIJFILE" fatal error message if a database data file (JOBS.RDA, JOBS.SNP) is specified instead of a database after image journal file in an RMU/RECOVER command. The last RMU/RECOVER command shows that if the after image journal file specified has an invalid format but is not a database root or data file, only the "%RMU–F–BADAIJFILE" message is output.

```
$!
$! Database root file specified instead of an AIJ file
$!
$ rmu/recover device:[directory]mf_personnel.rdb
%RMU-F-BADAIJFILE, this file is not a valid after image journal file,
please correct the file specification
-RMU-F-DBROOTFILE, specify a database after image journal file, this
is a database root file
%RMU-F-FTL_RCV, Fatal error for RECOVER operation at 25-JUL-2017
10:02:53.28
$!
$! Database data files specified instead of an AIJ file
$!
```

```

$ rmu/recover device:[directory]jobs.rda
%RMU-F-BADAIJFILE, this file is not a valid after image journal file,
please correct the file specification
-RMU-F-DBDATAFILE, specify a database after image journal file, this
is a database data file
%RMU-F-FTL_RCV, Fatal error for RECOVER operation at 25-JUL-2017
10:04:40.22
$!
$ rmu/recover device:[directory]jobs.snp
%RMU-F-BADAIJFILE, this file is not a valid after image journal file,
please correct the file specification
-RMU-F-DBDATAFILE, specify a database after image journal file, this
is a database data file
%RMU-F-FTL_RCV, Fatal error for RECOVER operation at 25-JUL-2017
10:07:45.16
$!
$! Invalid database AIJ file specified
$!
$ rmu/recover device:[directory]invalid.aij
%RMU-F-BADAIJFILE, this file is not a valid after image journal file,
please correct the file specification
%RMU-F-FTL_RCV, Fatal error for RECOVER operation at 25-JUL-2017
10:09::32.16

```

7.1.7 RMU Extract Now Outputs ALTER DATABASE For Storage Area Access Mode

This release of RMU Extract allows the output of the READ WRITE or READ ONLY clause for the storage area access mode. By default, a comment is written to the script documenting the current mode. Now, using the ACCESS_MODE keyword for the DEFAULTS qualifier will output an ALTER DATABASE (CHANGE DATABASE for RDO) statement that modifies the access mode for all storage areas (except for RDB\$SYSTEM). If the keyword is negated (NOACCESS_MODE), then the comment is not written to the SQL or RDO script.

The following example shows using the new keyword.

```

$          RMU/EXTRACT/ITEM=DATABASE -
          /DEFAULT=(ACCESS_MODE) -
          /OPTION=(NOHEADER,FILENAME_ONLY) -
          MF_PERSONNEL -
          /OUTPUT=DB.SQL
.
.
.

```

7.1.8 RMU/RECOVER RMU-F-BACKUPNOAIJ, RMU-F-TSNNOSYNC, RMU-F-CANTSYNCTSNS Error Messages

For the the recovery of Oracle Rdb databases from After Image Journal (AIJ) files using the RMU/RECOVER command, Transaction Sequence Number (TSN) values are maintained in the database root file and in the open record and transaction records of each journal file. Each TSN

number represents a database transaction which modified the database. The highest committed TSN number in the database root file determines where in the journal file or backed-up or optimized journal file RMU/RECOVER will start the roll forward operation. An AIJ file will only be applied to the database if the TSN number in the open record of the AIJ file is less than or equal to the highest committed TSN number in the database root file. Individual transactions contained in an AIJ file are ignored until the TSN of an individual transaction equals the highest committed TSN number in the database root file.

If the TSN number in the open record of an AIJ file is greater than the highest committed TSN number in the database root file, none of the transactions in the AIJ file will be recovered since there are missing transactions that need to be recovered before the transactions that are contained in the current AIJ file are recovered to prevent loss of data and database corruption. Previously, if the TSN number in the open record of an AIJ file was greater than the highest committed TSN number in the database root file, RMU/RECOVER would read through the entire journal file, ignoring all transactions because the TSN values of the individual transaction records in the AIJ file are all greater than the highest committed TSN number in the database root file. After ignoring all transactions, RMU/RECOVER would put out the following warning message.

```
%RMU-W-NOTRANAPP, no transactions in this journal were applied
```

Now, if at the start of the RMU/RECOVER operation, the TSN number in the open record of the first AIJ file to be processed is greater than the highest committed TSN number in the database root file, the recovery operation will be immediately aborted to avoid reading through the entire AIJ file and any additional AIJ files to be processed, ignoring all transactions. The recovery operation is aborted based on the open record of the first AIJ file to be processed since all AIJ files processed after the first AIJ file should also have TSN numbers in their open records which are greater than the highest committed TSN number in the database root file because all transactions must be recovered in the correct original sequence to prevent loss of data and database corruption.

When the RMU/RECOVER operation is aborted because at the start of the recover operation the TSN number in the open record of the first AIJ file to be processed is greater than the highest committed TSN number in the database root file, one of the following two fatal message sequences will be output.

```
%RMU-F-BACKUPNOAIJ, After Image Journaling was enabled after the
database was backed up or has since been disabled and reinitialized
-RMU-F-CANTSYNCTSNS, Last committed TSN 96 in the after image journal
file exceeds last committed TSN 35 in the database root
```

This message sequence is output by RMU/RECOVER if the database is either backed up before AIJ journaling is enabled and any after image journal files are defined for the database or if the database is backed up after AIJ journaling has been disabled and the after image journal state has not been reenabled and recovered by the database restore operation. A backup of the database should be made whenever changes are made to the database, prior to the database recovery which causes the following message to be output.

```
%RMU-W-DOFULLBCK, full database backup should be done to ensure future
recovery
```

If the database was backed up subsequent to after image journaling being enabled and after image journal files being defined, restore the database from that backup file and retry the recovery operation.

Oracle® Rdb for OpenVMS

```
%RMU-F-TSNNOSYNC, The transactions in this journal file are not
consistent with the transactions in this database root file
-RMU-F-CANTSYNCTSNS, Last committed TSN 448 in the after image journal
file exceeds last committed TSN 0 in the database root
```

This message sequence is output by RMU/RECOVER if unjournalled modifications were made to the database, or a copy of the database if the /ROOT qualifier was specified, that were not journalled. If the previous changes are contained in another after image journal file, that AIJ file should be applied before this AIJ file is applied. This message may also be output if the RMU/REPAIR/INITIALIZE=TSN command was executed to initialize the TSNs after the database was backed up. A full backup of the database should be made after any RMU/REPAIR operation is executed or any changes are made to the database which cause the following message to be output.

```
%RMU-W-DOFULLBCK, full database backup should be done to ensure future
recovery
```

Make sure that all previous changes made to the database have been journalled and that all after image journal files containing those changes are specified in the RMU/RECOVER command in the correct sequence order and applied to the correct database or copy of the database.

In the the following example, the fatal %RMU-F-BACKUPNOAIJ and -RMU-F-CANTSYNCTSNS messages are output by RMU/RECOVER because the database is backed up before after image journaling is enabled and any after image journal files are defined for the database. If the database is instead backed up immediately after the %RDMS-W-DOFULLBCK message is output, the RMU/RECOVER operation will succeed.

```
$ SQL
create database filename device:[directory]foo.rdb;
create table t1 (f1 int);
create unique index i1 on t1(f1);
commit;
disconnect all;
exit
$ RMU/BACKUP/NOLOG device:[directory]FOO.RDB BAR.RBF
$ SQL
alter database filename device:[directory]foo.rdb
journal is enabled
add journal foo file 'device:[directory]foo.aij';
%RDMS-W-DOFULLBCK, full database backup should be done to ensure
future recovery
exit
$ SQL
attach 'filename device:[directory]foo.rdb';
insert into t1 values (1);
1 row inserted
insert into t1 values (2);
1 row inserted
insert into t1 values (10);
1 row inserted
commit;
exit
$ SQL
drop database filename foo.rdb;
exit
$ RMU/RESTORE/NOCCD/NOLOG BAR.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.
```

Oracle® Rdb for OpenVMS

```
$ RMU/RECOVER/LOG/TRACE FOO.AIJ
%RMU-I-LOGRECDB, recovering database file device:[directory]FOO.RDB;1
%RMU-F-BACKUPNOAIJ, After Image Journaling was enabled after the database was
  backed up or has since been disabled and reinitialized
-RMU-F-CANTSYNCTSNS, Last committed TSN 96 in the after image journal file
  exceeds last committed TSN 35 in the database root
%RMU-F-FTL_RCV, Fatal error for RECOVER operation at 20-SEP-2017 15:02:57.15
```

In the the following example, the fatal %RMU-F-TSNNOSYNC and -RMU-F-CANTSYNCTSNS messages are output by RMU/RECOVER because the previous unjournalled RMU/REPAIR operation initializes the TSN values in the database root before the RMU/RECOVER command is executed and because the %RMU-I-FULBACREQ message calling for a database backup immediately after the RMU/REPAIR command is executed (which initializes the database root TSN values), is ignored.

```
$ sql
  alter database filename device:[directory]mf_personnel.rdb
  journal filename device:[directory]pers_aij.aij;
%RDMS-W-DOFULLBCK, full database backup should be done to ensure
  future recovery
  exit
$!
$! Create entry in aij file
$!
$ sql
  attach 'filename device:[directory]mf_personnel.rdb';
  update employees
  set address_data_1 = '10 Ridge St.'
  where employee_id = '00164';
1 row updated
  commit;
  disconnect all;
  exit;
$!
$! Re-set TSNS
$ rmu/repair/initialize=tsns device:[directory]mf_personnel.rdb
%RMU-I-AIJ_ENABLED, This database has after image journaling enabled...
  You should create a new journal after this operation completes.
%RMU-I-FULBACREQ, A full backup of this database should be performed after
  RMU REPAIR
$!
$! Try to apply original .aij; should not succeed
$!
$ rmu/recover/log/root=device:[directory]mf_personnel.rdb
device:[directory]pers_aij.aij
%RMU-I-LOGRECDB, recovering database file DISK:[DIRECTORY]MF_PERSONNEL.RDB;2
%RMU-F-TSNNOSYNC, The transactions in this journal file are not consistent
  with the transactions in this database root file
-RMU-F-CANTSYNCTSNS, Last committed TSN 448 in the after image journal file
  exceeds last committed TSN 0 in the database root
%RMU-F-FTL_RCV, Fatal error for RECOVER operation at 20-SEP-2017 16:08:16.96
```

7.1.9 Delimited_Text Keywords Can Now Be Negated For RMU Load And Unload

In previous versions of Oracle Rdb, the RMU Load and RMU Unload command keywords PREFIX, SUFFIX, NULL and SEPARATOR could be specified as empty quoted strings. This notation was

used to eliminate one or more of these options from the delimited text output.

In this release, these keywords can be negated (NOPREFIX, NOSUFFIX, NONULL, NOSEPARATOR) as required to achieve the same effect. This applies to the RMU Load, RMU Unload, and RMU Unload After_Journal commands. The output of any plan file will contain the empty strings if NOPREFIX, NOSUFFIX, or NOSEPARATOR is used.

The following example compares the empty string syntax with the negated keyword usage.

```
$ rmu/unload-
  /record=(nofile,format=delimited,prefix=" ",suffix=" ",null="*") -
  db$:mf_personnel -
  work_status -
  sys$output:
0,INACTIVE,RECORD EXPIRED
1,ACTIVE ,FULL TIME
2,ACTIVE ,PART TIME
*,*,*
%RMU-I-DATRECUNL, 4 data records unloaded 28-MAR-2017 17:24:07.43.
$ rmu/unload-
  /record=(nofile,format=delimited,noprefix,nosuffix,null="*") -
  db$:mf_personnel -
  work_status -
  sys$output:
0,INACTIVE,RECORD EXPIRED
1,ACTIVE ,FULL TIME
2,ACTIVE ,PART TIME
*,*,*
%RMU-I-DATRECUNL, 4 data records unloaded 28-MAR-2017 17:24:07.56.
$
```

7.1.10 RMU Load Now Supports User Defined Conversion Routines

This release of Oracle Rdb adds a new column attribute to the record definition file syntax. The new STORE USING clause allows the record definition file (RRD) to indicate to RMU how to store the data in the target column by specifying the name of a transformation function. This function may be a SQL or an external function existing in the target database.

For example, when a delimited data file is read, there may exist column values in formats not acceptable to Oracle Rdb. The routine specified by the STORE USING clause allows the source value from the data file to be manipulated prior to being inserted into the table.

Some examples include:

- ◆ Date formats that only specify two digit year, which requires century be derived by some rule (year < 50 means adding 1900, otherwise they are assumed to add 2000)
- ◆ Date/time values that specify fractional seconds precision larger than supported by SQL
- ◆ Numeric decimal marker and digit group separators different from those implicitly supported by Oracle Rdb. For example, a value might be saved by an application in some countries as 1.234.567,89 which would not be accepted by Rdb without some transformation.
- ◆ Character fields might have leading or trailing spaces, but the database column expects them

to be trimmed.

For this release, only functions that accept one argument are supported from the STORE USING clause.

The examples directory SQL\$SAMPLE includes a module definition that provides example routines that transform source text prior to insert into the target column. This script is provided for use by the database programmer, as well as to be used as a model for locally created routines. The script, CVT_MODULE.SQL, contains the following routines:

- ◆ CONVERT_DATE_VMS_7 accepts a text string which represents a date VMS string with 7 digits of fractional second value.
- ◆ CONVERT_DECIMAL_MARK converts numeric values by removing any digit group separators and changing the decimal mark to ".". It then returns a string for implicit conversion to the columns data type.
- ◆ CONVERT_DATE_DD_MON_YY accepts a text string with the month as a three letter (English) abbreviation, and a 2 or 4 digit year. It parses the date and returns a DATE ANSI value.
- ◆ SET_NULL_WHEN_ZERO. In some cases special values in the data file represent an UNKNOWN state. This script assumes that zero equates to NULL. An example of this might be SALARY_AMOUNT in the SALARY_HISTORY table.

In some cases, transformations can be solved by using SQL builtin functions. The following are directly supported and will be called with one parameter defaulting the optional parameters; TRIM, RTRIM, LTRIM, UPPER, LOWER and SQRT.

The routine name, (unless it is a SQL builtin), must represent an SQL or External routine in the database. RMU performs some simple validation but it is expected that the input parameter and result type are compatible with the data type of the RMU Load source and the target column data type.

The following example shows a simple routine to filter the decimal mark in an input field.

```
create module EXAMPLE
function CONVERT_DECIMAL_MARK (in :v varchar(40))
returns varchar(40)
comment is 'Only preserve numbers and sign from the input'
/
'and substitute decimal mark';
return translate (:v, '+-0123456789,. ', '+-0123456789. ');
end module;
```

Each field in the record definition file may have at most one STORE USING clause.

The following example shows a simple record definition file that would be used to load data from a delimited data file.

```
DEFINE FIELD EMPLOYEE_ID DATATYPE IS TEXT SIZE IS 5.
DEFINE FIELD SALARY_AMOUNT DATATYPE IS TEXT SIZE IS 30.
DEFINE FIELD SALARY_START DATATYPE IS TEXT SIZE IS 10.
DEFINE FIELD SALARY_END DATATYPE IS TEXT SIZE IS 10.
DEFINE RECORD SALARY_HISTORY.
EMPLOYEE_ID .
SALARY_AMOUNT store using CONVERT_DECIMAL_MARK.
SALARY_START .
SALARY_END .
END SALARY_HISTORY RECORD.
```

7.1.11 New CARDINALITY Option for SHOW TABLE Command

This release of Oracle Rdb adds support for the CARDINALITY option to SHOW TABLE and enhances the support for SHOW INDEX. The CARDINALITY option (unlike other SHOW options) adds the display of cardinality information to other table and index displays.

- ◆ SHOW INDEX (CARDINALITY)

In prior versions, this option would only display output for index column cardinality if the index was not unique. SQL now displays the table's approximate cardinality for unique indices.

Index column cardinality is not maintained for the last index column, as this is the same value as the index cardinality.

- ◆ SHOW TABLE (CARDINALITY)

This command now also displays the approximate cardinality as recorded in the Rdb system tables for the named tables and their indices.

The following example shows the additional output when the CARDINALITY option is used.

```
SQL> show table (cardinality,index) salary_history;
Information for table SALARY_HISTORY
```

Table cardinality: 729

Indexes on table SALARY_HISTORY:

SH_EMPLOYEE_ID with column EMPLOYEE_ID

Index cardinality: 100

Duplicates are allowed

Type is Sorted

Key suffix compression is DISABLED

Node size 430

Percent fill 70

7.1.12 New CONSTRAINT Naming for Domain Constraints

This release of Oracle Rdb supports the naming of domain constraints. The CREATE and ALTER DOMAIN Statements now allow the CHECK constraint to be named. The name can later be used by the ALTER DOMAIN ... DROP CONSTRAINT statement.

Syntax for CREATE DOMAIN Statement

```
domain-constraint =
+-----+-----+-----+-----+-----+-----+-----+-----+
|-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |                                     |
+--> CONSTRAINT <constraint-name> --+                                     |
|                                     |                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |                                     |
+--> constraint-attributes --+                                     |
|                                     |                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

Usage Notes for CREATE DOMAIN Statement

- ◆ The optional CONSTRAINT clause is used to give a name to a domain constraint. This name must not be the same as an existing domain, table or column constraint nor that of a view WITH CHECK OPTION.
- ◆ The CONSTRAINT name is a simple identifier. It cannot be qualified by an alias as it is not a separate database object.
- ◆ The CONSTRAINT name can be used in a subsequent ALTER DOMAIN ... DROP CONSTRAINT clause.

Syntax for ALTER DOMAIN Statement

```
alter-domain-constraints =
---+--> domain-constraint -----+-->
|
|
+--> DROP CONSTRAINT <constraint-name> ---+
|
|
+--> DROP ALL CONSTRAINTS -----+

```

Additional Usage Notes for ALTER DOMAIN Statement

- ◆ The ALTER DOMAIN ... ADD CONSTRAINT clause performs an implicit ALTER DOMAIN ... DROP ALL CONSTRAINTS prior to applying the new constraint to the domain. Any constraint name defined by prior statements will also be dropped.

7.1.13 New AS Result-type Clause for CREATE SEQUENCE Statement

This release of Oracle Rdb supports the ANSI and ISO SQL Database Language AS clause for CREATE SEQUENCE. The AS clause specifies a data type which will be returned by the sequence. Oracle Rdb restricts the result to unscaled integer types: TINYINT, SMALLINT, INTEGER (INT) and BIGINT (QUADWORD). Unless specified by the CREATE SEQUENCE Statement, SQL will implicitly set the MAXVALUE or the MINVALUE to the extreme values that can be stored in such a data type.

The following example shows the effect of the AS clause.

```
SQL> create sequence new_departments as integer cycle;
SQL> show sequence new_departments;
      NEW_DEPARTMENTS
Sequence Id: 1
Initial Value: 1
Minimum Value: 1
Maximum Value: 2147483647
Next Sequence Value: 1
Increment by: 1
Cache Size: 20
No Order
Cycle
No Randomize
Wait
SQL>
```

7.1.14 New GENERATED Column Support

This release of Oracle Rdb adds support for the ANSI/ISO SQL Database Language Standard clause GENERATED ALWAYS.

The following new clauses are supported:

- ◆ GENERATED ALWAYS AS IDENTITY [*identity-attributes*]

This clause is equivalent to the Oracle Rdb syntax **IDENTITY** [*identity-attributes*] and is added for compatibility with Oracle Database and the ANSI/ISO SQL Database Language Standard.

When defining a column, the data type of the column can be provided, as shown in the following example, and will result in an implicit CAST of the value-expression to that data type (or domain).

```
SQL> create domain SEQ_NO_DOM integer;
SQL>
SQL> create table SAMPLE
cont>      (seq_no SEQ_NO_DOM generated always as identity
cont>      !...
cont>      );
SQL>
SQL> show table (column) SAMPLE;
Information for table SAMPLE

Columns for table SAMPLE:
Column Name                Data Type                Domain
-----
SEQ_NO                      INTEGER
  Computed:      Generated always as Identity
  !...
```

If the data type is omitted, then the default will be BIGINT.

- ◆ GENERATED BY DEFAULT AS IDENTITY [*identity-attributes*]

This clause is similar to the GENERATED ALWAYS AS IDENTITY clause, with the exception that the application programmer may INSERT a value instead of having Oracle Rdb compute and store a value.

In contrast, the GENERATED ALWAYS clause is treated as a read-only column. Note: the database administrator can also use the SET FLAGS 'AUTO_OVERRIDE' statement to temporarily treat GENERATED ALWAYS columns as GENERATED BY DEFAULT columns.

If an explicit value is inserted by the application, then it is possible that the sequence associated with the IDENTITY column will generate a duplicate value. The application must be prepared to handle this case and Oracle Rdb cannot guarantee uniqueness of values in this column.

- ◆ GENERATED ALWAYS AS (*value-expression*)

This clause is equivalent to the Oracle Rdb syntax **AUTOMATIC INSERT AS** *value-expression* and is added for compatibility with Oracle Database and the ANSI/ISO SQL Database Language Standard.

When defining a column, the data type of the column can be provided, as shown in the following example, and will result in an implicit CAST of the value-expression to that data

type (or domain).

```
SQL> create table SAMPLE
cont>      (seq_no SEQ_NO_DOM generated always as identity
cont>      ,row_ts timestamp(2) generated always as ( current_timestamp )
cont>      !...
cont>      );
SQL>
SQL> show table (column) SAMPLE;
Information for table SAMPLE

Columns for table SAMPLE:
Column Name                Data Type                Domain
-----
SEQ_NO                      INTEGER
  Computed:                Generated always as Identity
ROW_TS                      TIMESTAMP(2)
  Computed:                Generated always as ( current_timestamp )
  !...

SQL>
```

If the column data type is not specified, then the data type will be derived from the value expression.

◆ **GENERATED BY DEFAULT AS (value-expression)**

This clause is similar to the **GENERATED ALWAYS** clause, with the exception that the application programmer may **INSERT** a value instead of having Oracle Rdb compute and store a value.

In contrast, the **GENERATED ALWAYS** clause is treated as a read-only column. Note: the database administrator can also use the **SET FLAGS 'AUTO_OVERRIDE'** statement to temporarily treat **GENERATED ALWAYS** columns as **GENERATED BY DEFAULT** columns.

7.1.15 Enhancements to **INCLUDE** Statement

Bug 25910172

This release of Oracle Rdb enhances the **INCLUDE** statement for the SQL Precompiler. The **INCLUDE** statement can now include modules from a referenced text library.

◆ **INCLUDE MODULE <modulename> FROM LIBRARY <library-file-spec>**

This command will include the source text from the named text library. The text library should be created using the OpenVMS command **LIBRARY/CREATE/TEXT**. The name of the modules in that library can be determined using the **LIBRARY/LIST/TEXT**. It is possible that these modules are specifically named using the **/MODULE** qualifier on the **LIBRARY** command.

```
$ LIBRARY/CREATE/TEXT PERSONNEL_DEFS.TLB
$ LIBRARY/REPLACE/TEXT PERSONNEL_DEFS.TLB EMPS.LIB/MODULE=EMPLOYEES_REC
$ LIBRARY/REPLACE/TEXT PERSONNEL_DEFS.TLB SH.LIB/MODULE=SALARY_HISTORY_REC
$ LIBRARY/REPLACE/TEXT PERSONNEL_DEFS.TLB JH.LIB/MODULE=JOB_HISTORY_REC
$
```

To reference this text library, the application would use an INCLUDE statement as shown below:

```
EXEC SQL INCLUDE MODULE EMPLOYEES_REC FROM LIBRARY 'PERSONNEL_DEFS.TLB'
END-EXEC
```

◆ INCLUDE MODULE <modulename>

This abbreviated statement defaults to using the text library named SQL\$TEXT_LIBRARY in the default directory, or referenced by the logical name SQL\$TEXT_LIBRARY.

```
EXEC SQL INCLUDE MODULE EMPLOYEES_REC
END-EXEC
```

Usage Notes

- ◆ Using the INCLUDE command makes any included text visible to the SQL Precompiler and also the target language. Use this command when you wish to make variable and record definitions visible to SQL or if the included text also contains EXEC SQL directives.
- ◆ The module included from a text library may not also include the INCLUDE file-spec statement nor the INCLUDE MODULE statement.
- ◆ The default file type for the LIBRARY is .TLB
- ◆ If the text library is created with case sensitive names, then the MODULE name must be in quotes to preserve the case of the name.

```
$ LIBRARY/CREATE=CASE_SENSITIVE:yes/TEXT PERSONNEL_DEFS.TLB
$ LIBRARY/REPLACE/TEXT -
  PERSONNEL_DEFS.TLB -
  JH.LIB/MODULE="JobHistoryRecord"
```

In such cases, the SQL\$PRE command line, or the MODULE header must specify that QUOTING RULES are enabled to allow quoted names. This can be specified using /SQLOPTIONS qualifier to specify either ANSI_IDENTIFIERS or ANSI_QUOTING, or compiling with a DECLARE MODULE statement in a context file.

```
$          ! Use a context file and set SQL99 quoting rules
$          CREATE CONTEXT_FILE.SQL
declare module TESTING
  pragma (ident 'v1.00')
  quoting rules sql99;
$          DEFINE/USER SQL$TEXT_LIBRARY INCLUDE_MODULE.TLB
$          SQL$PRE/COBOL SAMPLE_APP CONTEXT_FILE.SQL
```

7.1.16 New Support for DEFAULT Index NODE SIZE Calculation

Bug 27484661

Prior to Oracle Rdb V7.3.1, the default node size was computed as 430, or when that value was too small for longer keys, 860. The problem with these sizes were that they did not fill a complete page. So there remained wasted space on a page that could not be used.

As part of a redesign of this functionality, it was decided that since the user did not define a NODE

SIZE, it was beneficial for most INSERT and query environments to use the maximum node size that could be stored on a page.

Some environments would prefer the default NODE SIZE for a sorted index to be smaller than that currently computed by the Rdb. This may be due to activity (DELETE and UPDATE of key values), concurrency where the application wants fewer keys to be locked, or when the page size is very large, say 32 to 63 blocks.

In this release, the logical name RDMS\$DEFAULT_INDEX_NODE_SIZE_SMALL can be defined to enable an alternate node size computation. The node sizes are still space filling on the page but are similar in size to that of prior versions.

In addition, the RDMS\$SET_FLAGS logical name or SET FLAGS statement can specify 'INDEX_SIZING(SMALL)' to select this algorithm. The setting 'INDEX_SIZING(LARGE)' or 'NOINDEX_SIZING' will revert to the other algorithm.

Oracle recommends that applications choose node sizes applicable to their application needs. These defaults are for ease of use and may not be best for all environments, key data values or application activity.

7.1.17 New LANGUAGE Support From RMU Extract Command

This release of Oracle Rdb supports extracting record definitions for selected or all tables in a database. The Language qualifier will now support the languages; CC, COBOL and Pascal.

◆ CC

For the C language, you must specify CC to avoid ambiguity with an abbreviated COBOL language.

The output for C is a *typedef* with the columns of the table and named with the table name. This definition is designed to be used by the SQL Precompiler and so includes pseudo data types and CHARACTER SET clauses to establish the correct semantics for the precompiler. The following example shows the output for the WORK_STATUS table. Note that, by default, RMU Extract assumes null terminated strings and adds one to the length. This can be disabled by specifying NONULL_TERMINATED for the Option qualifier.

Applications would then declare a variable using this typedef definition as shown in the following code fragment.

```
$ rmu/extract/item=table /lang=cc sql$database/option=(noheader,
match:work_status,audit)

/* Table: WORK_STATUS (null terminated)
// Created on 11-JUL-2018 17:39:59.20
// Last altered on 11-JUL-2018 17:39:59.21
// Created by HR_SERVICES
//
*/

#ifdef _WORK_STATUS_
#define _WORK_STATUS_
typedef struct {
    char STATUS_CODE[2];
```


Oracle® Rdb for OpenVMS

```
        char STATUS_NAME[9];
        char STATUS_TYPE[15];
    } WORK_STATUS;
#endif // WORK_STATUS
...
exec sql
    include 'work_status.h'

WORK_STATUS work_status_rec;

exec sql
    fetch work_status_cursor into :work_status_rec;
```

Note that the generated text file must be included by the EXEC SQL INCLUDE or EXEC SQL INCLUDE MODULE ... FROM LIBRARY statements so that the definition is visible to the SQL Precompiler.

◆ COBOL

The output for COBOL is a 01 data division definition with the columns of the table and named with the table name. VARCHAR and VARBINARY columns are represented by level 49 field with two subfields to represent the length (LEN) and body (VAL) of the string. This definition is designed to be used by the SQL Precompiler and so includes pseudo data types and CHARACTER SET clauses to establish the correct semantics for the precompiler. The following example shows the representation of a VARCHAR column.

```
$ rmu/extract -
  /item=table -
  /language=cobol -
  PERSONNEL -
  /option=(noheader,audit,match:candidates)
...
** Table: CANDIDATES
** Created on 11-JUL-2018 17:39:59.03
** Never altered
** Created by HR_SERVICES
**

01 CANDIDATES.
   05 LAST_NAME                picture X(14).
   05 FIRST_NAME               picture X(10).
   05 MIDDLE_INITIAL          picture X(1).
   05 CANDIDATE_STATUS        .
       49 LEN                  picture S9(4) comp.
       49 VAL                  picture X(255).
```

Note that the generated text file must be included by the EXEC SQL INCLUDE or EXEC SQL INCLUDE MODULE ... FROM LIBRARY statements so that the definition is visible to the SQL Precompiler.

◆ Pascal

The output for Pascal is a *type* with the columns of the table and named with the table name. This definition is designed to be used by the SQL Precompiler and so includes pseudo data types and CHARACTER SET clauses to establish the correct semantics for the precompiler. Applications would then declare a variable (VAR) using this type definition as shown in the following code fragment.

```
$ rmu/extract/item=table -
```

```

/lang=Pascal -
/output=work_status.p -
sql$database -
/option=(noheader,match:work_status,audit)
.
.
.
exec sql
    include 'work_status.p';

var
    work_status_rec : work_status := ZERO;
.
.
.

```

Note that the generated text file must be included by the EXEC SQL INCLUDE or EXEC SQL INCLUDE MODULE ... FROM LIBRARY statements so that the definition is visible to the SQL Precompiler.

7.1.18 Enhancements for CREATE and ALTER MODULE Statements

With this release of Oracle Rdb, the following enhancements have been made to the module and routine functionality.

1. The module attributes can now appear in any order.
In prior releases, the clauses STORED NAME IS, LANGUAGE SQL, AUTHORIZATION, and COMMENT IS were required to appear in that order, although any and all could be omitted. SQL now allows these and new clauses to be in any order following the name of the module and before the DECLARE statements.
2. A new EXTERNAL DEFAULTS clause has been added to the module header.
In prior releases, each external routine within the module had to specify the LOCATION, LANGUAGE, PARAMETER STYLE, BIND ... SITE, and BIND SCOPE clauses. In many cases, the module referenced just one shareable image and all these values were the same but were duplicated for every external routine.
For example, these three routines are defined in the OpenVMS runtime library.

```

SQL> create module DCL_SYMBOLS
cont>
cont>   procedure LIB$SET_SYMBOL (in :symbol varchar(40) by descriptor,
cont>                           in :value_string varchar(40) by descriptor,
cont>                           in :table_type_indicator integer = 2);
cont>     external name LIB$SET_SYMBOL
cont>               location 'SYS$SHARE:LIBRTL.EXE'
cont>               language GENERAL
cont>               parameter style GENERAL;
cont>
cont>   procedure LIB$GET_SYMBOL (in :symbol varchar(40) by descriptor,
cont>                             out :resultant_string varchar(40) by descriptor,
cont>                             out :resultant_length smallint,
cont>                             in :table_type_indicator integer = 2);
cont>     external name LIB$GET_SYMBOL
cont>               location 'SYS$SHARE:LIBRTL.EXE'
cont>               language GENERAL

```

Oracle® Rdb for OpenVMS

```
cont>          parameter style GENERAL;
cont>
cont> procedure LIB$DELETE_SYMBOL (in :symbol varchar(40) by descriptor,
cont>                               in :table_type_indicator integer = 2);
cont>     external name LIB$DELETE_SYMBOL
cont>         location 'SYS$SHARE:LIBRTL.EXE'
cont>         language GENERAL
cont>         parameter style GENERAL;
cont>
cont> end module;
```

With this release of Oracle Rdb, many of these attributes can be specified once in the module header as part of the EXTERNAL DEFAULTS clause and omitted for procedure and function definitions. Often the only clause required for an external routine is the EXTERNAL keyword.

As can be seen in this example, this simplifies the definition. In addition, the name of the external routine body matches the name within SQL so the NAME clause is also omitted.

```
SQL> create module DCL_SYMBOLS
cont>     external defaults (
cont>         location 'SYS$SHARE:LIBRTL.EXE'
cont>         language GENERAL
cont>         parameter style GENERAL
cont>     )
cont>
cont> procedure LIB$SET_SYMBOL (in :symbol varchar(40) by descriptor,
cont>                             in :value_string varchar(40) by descriptor,
cont>                             in :table_type_indicator integer = 2);
cont>     external;
cont>
cont> procedure LIB$GET_SYMBOL (in :symbol varchar(40) by descriptor,
cont>                             out :resultant_string varchar(40) by descriptor,
cont>                             out :resultant_length smallint,
cont>                             in :table_type_indicator integer = 2);
cont>     external;
cont>
cont> procedure LIB$DELETE_SYMBOL (in :symbol varchar(40) by descriptor,
cont>                             in :table_type_indicator integer = 2);
cont>     external;
cont>
cont> end module;
```

3. The ALTER MODULE statement has been enhanced to also support the EXTERNAL DEFAULTS created with the module. Therefore, any external routine added to the module with the ADD FUNCTION or ADD PROCEDURE clauses will inherit any missing attributes from the original module definitions.

For example, assume this modified example where the routine LIB\$DELETE_SYMBOL is added later using the ALTER MODULE statement.

```
SQL> create module DCL_SYMBOLS
cont>     external defaults (
cont>         location 'SYS$SHARE:LIBRTL.EXE'
cont>         language GENERAL
cont>         parameter style GENERAL
cont>     )
cont>
cont> procedure LIB$SET_SYMBOL (in :symbol varchar(40) by descriptor,
cont>                             in :value_string varchar(40) by descriptor,
cont>                             in :table_type_indicator integer = 2);
```

```

cont>      external;
cont>
cont>      procedure LIB$GET_SYMBOL (in :symbol varchar(40) by descriptor,
cont>                                out :resultant_string varchar(40) by descriptor,
cont>                                out :resultant_length smallint,
cont>                                in :table_type_indicator integer = 2);
cont>      external;
cont>
cont> end module;
    
```

At a later time, ALTER MODULE can be executed to add other routines in the same shareable image.

```

SQL> ! Add new routine using ALTER DATABASE
SQL> !
SQL> alter module DCL_SYMBOLS
cont>      add
cont>      procedure LIB$DELETE_SYMBOL (in :symbol varchar(40) by descriptor,
cont>                                    in :table_type_indicator integer = 2);
cont>      external;
cont> end module;
    
```

7.1.19 New RMU Dump Symbols Command

Enhancement Bugs 804046 and 2790736

This release of Oracle Rdb adds a new RMU command: RMU Dump Symbols.

RMU Dump Symbols Command

Displays or writes to a specified output file the contents of database root file information. The output is similar to that from RMU Dump Header except that the output is in the form of a DCL command procedure that defines global DCL symbols.

Syntax

RMU/Dump/Symbols root-file-spec

Command Qualifiers

```

/Execute
/Output[=file-spec]
/[No]Prefix[=text-string]
    
```

Defaults

```

NoExecute
/Output=SYS$OUTPUT
/Prefix=RMU$
    
```

Description

This command is designed for database administrators who wish to write DCL procedures that react to the current state of the Rdb database.

This command does not open the database and only provides access to the static on-disk database root file.

Command Parameters

- ◆ `root-file-spec`
A file specification for the database root file whose root file header information you want to display.

Qualifiers

- ◆ `Execute`
The file created by the `/OUTPUT` qualifier is executed before returning control to DCL. This allows immediate use of the defined global symbols. The default is `NoExecute`.
- ◆ `Output[=file-spec]`
The name of the output command procedure created by RMU. The default is `SYSS$OUTPUT` (which cannot be used by `/EXECUTE`).
- ◆ `Prefix=test-string`
`NoPrefix`
By default, the generated symbol names start with `RMU$`. However, the database administrator can replace this with any text string. This would allow, for instance, two `RMU Dump Symbol` commands to be executed on different databases and the generated (unique) symbols compared.
If `NoPrefix` is used then no prefix string is added to the created DCL symbols.

The following example shows a simple command procedure to get the `CLIENT_FULL_BACKUP_TIMESTAMP` for the named database and compute the delta time since it was last backed up.

Example 7-5 Using RMU Dump Symbols

```

$ v = 'f$verify(0)
$ set noon
$!
$! Check the last full backup date and see if backup is past due
$!
$ temp_file = "temp" + f$getjpi(0,"PID") + ".tmp;"
$ RMU/DUMP/SYMBOL/EXECUTE/PREFIX=PERS_/OUTPUT=&TEMP_FILE SQL$DATABASE
$ delta_time = f$delta_time (PERS_CLIENT_FULL_BACKUP_TIMESTAMP,"TODAY")
$ days = f$integer(f$element(0," ",delta_time))
$ if days .gt. 7
$ then
$     alert_text = f$fao("Database PERS not backed up in !SL day!%S", days)
$     write sys$output alert_text
$     ! reply/username=DBADMIN "'alert_text'"
$ endif
$ delete &temp_file
$ exit ! 'f$verify(v)'

```

7.1.20 New Options to SET SQLDA Statement

The `SET SQLDA` Statement now supports the `ENABLE` and `DISABLE` of `TRUNCATE WARNINGS`. The default behavior when `SET DIALECT` establishes the dialect as `SQL92`, `SQL99`,

SQL2011, or an ORACLE dialect is to generate an error when an assignment would cause a string value to be truncated.

Note

Trailing spaces characters are ignored when determining if a string is truncated.

This setting of the SQLDA allows dynamic applications to enable or disable this behavior for all dialects, including SQLV40 (default dialect) and SQL89.

```
enable-option =
--> FULL QUERY HEADER -----+-->
|                               |
+--> INSERT RETURNING -----+
|                               |
+--> INTEGER COUNT -----+
|                               |
+--> NAMED MARKERS -----+
|                               |
+--> NULL ELIMINATION WARNINGS -+
|                               |
+--> ROWID TYPE -----+
|                               |
+--> TRUNCATE WARNINGS -----+
```

The following example uses Dynamic SQL to execute various INSERT statements. The tool displays the error reported for string truncation.

```
--> CREATE TABLE SAMPLE_TABLE (COL1 CHAR);
--> INSERT INTO SAMPLE_TABLE VALUES ('xxx');
inputs: 0
-> !;
-> SET SQLDA 'enable truncate warnings';
inputs: 0
--> INSERT INTO SAMPLE_TABLE VALUES ('xxx');
inputs: 0
Error -306:
%RDB-E-TRUN_STORE, string truncated during assignment to a column
-> !;
-> SET SQLDA 'disable truncate warnings';
inputs: 0
--> INSERT INTO SAMPLE_TABLE VALUES ('xxx');
inputs: 0
```

7.1.21 More New Options to SET SQLDA Statement

The SET SQLDA Statement now supports the ENABLE and DISABLE of NULL ELIMINATION WARNINGS. The default behavior when SET DIALECT establishes the dialect as SQL92, SQL99, SQL2011, or an ORACLE dialect is to generate a warning when an aggregate function (COUNT, MIN, MAX, AVG, STDDEV, etc) eliminates NULL values when computing a result.

This setting of the SQLDA allows dynamic applications to enable or disable this behavior for all dialects, including SQLV40 (default dialect) and SQL89.

```

enable-option =
+--> FULL QUERY HEADER -----+-->
|
+--> INSERT RETURNING -----+
|
+--> INTEGER COUNT -----+
|
+--> NAMED MARKERS -----+
|
+--> NULL ELIMINATION WARNINGS -+
|
+--> ROWID TYPE -----+
|
+--> TRUNCATE WARNINGS -----+

```

The following example uses Dynamic SQL to execute a COUNT function on a column that has some values set to NULL. The tool displays the warning reported in such cases.

```

Enter statement:
SET DIALECT 'SQL99';
inputs: 0
Enter statement:
SELECT COUNT (MIDDLE_INITIAL) FROM EMPLOYEES;
inputs: 0
out: [0] typ=Bigint {505} len=8
--> reported warning; sqlcode=1003
[SQLDA - displaying 1 fields]
0/: 64
Enter statement:
SET SQLDA 'DISABLE NULL ELIMINATION WARNINGS';
inputs: 0
Enter statement:
SELECT COUNT (MIDDLE_INITIAL) FROM EMPLOYEES;
inputs: 0
out: [0] typ=Bigint {505} len=8
[SQLDA - displaying 1 fields]
0/: 64
Enter statement:

```

Chapter 8

Enhancements And Changes Provided in Oracle Rdb Release 7.3.2.1

8.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.2.1

8.1.1 Oracle Rdb 7.3.2.1 Certified on OpenVMS 8.4–2 from VMS Software Inc. and Integrity i4 systems from HPE

This version of Rdb has been certified to run on OpenVMS Version 8.4–2 from VMS Software Inc. and Integrity i4 systems from HPE.

8.1.2 RMU/SHOW AFTER_JOURNAL [NO]CHECKPOINT Qualifier

The RMU/SHOW AFTER_JOURNAL [NO]CHECKPOINT qualifier can be used to request that all active database processes on all nodes perform a checkpoint when the /BACKUP_CONTEXT qualifier is also specified to set After Image Journal database symbols based on the current After Image Journal configuration defined in the database root file and the database fast commit to journal feature is enabled for the database. For more information on these After Image Journal database symbols that begin with "RDM\$AIJ_", see the documentation for the /BACKUP_CONTEXT qualifier.

The checkpoint occurs immediately before the AIJ global symbols are defined or modified. The checkpoint will only be executed if the RMU/SHOW AFTER_JOURNAL command /BACKUP_CONTEXT qualifier is also specified and the database fast commit to journal feature is currently enabled.

The syntax for this qualifier is as follows:

```
/[NO]CHECKPOINT
```

The default if this qualifier is not specified is /NOCHECKPOINT.

The following example shows a database defined with circular After Image Journal files and fast commit to the journal files enabled. Then an RMU/SHOW AFTER_JOURNAL command is executed with the /BACKUP_CONTEXT qualifier and the /CHECKPOINT qualifier specified. In this case, a global checkpoint will be executed before the global symbols that start with 'RDM\$AIJ_' are defined or modified.

```
$ SQL
create database filename foo
  reserve 10 journals
  create storage area RDB$SYSTEM
    filename foo
    alloc 3000
    snap alloc 1;
create table tab (a int, b char(500)) ;
commit;
```

Oracle® Rdb for OpenVMS

```
disc all ;
alter database filename foo
  journal is enabled (fast commit enabled)
  add journal foo_aij_0
    filename test$scratch:foo_aij_0.aij
  add journal foo_aij_1
    filename test$scratch:foo_aij_1.aij
  add journal foo_aij_2
    filename test$scratch:foo_aij_2.aij
  add journal foo_aij_3
    filename test$scratch:foo_aij_3.aij
  add journal foo_aij_4
    filename test$scratch:foo_aij_4.aij
  add journal foo_aij_5
    filename test$scratch:foo_aij_5.aij
  add journal foo_aij_6
    filename test$scratch:foo_aij_6.aij
  add journal foo_aij_7
    filename test$scratch:foo_aij_7.aij
;
%RDMS-W-DOFULLBCK, full database backup should be done to ensure future recovery
exit
$ rmu/dump/header/out=foo.hdr foo
$ sear foo.hdr fast
  Fast Commit...
    - Fast commit is enabled
    - Fast incremental backup is enabled
$ rmu/show after_journal/backup_context/checkpoint foo
JOURNAL IS ENABLED -
  RESERVE 10 -
  ALLOCATION IS 512 -
  EXTENT IS 512 -
  OVERWRITE IS DISABLED -
  SHUTDOWN_TIMEOUT IS 60 -
  NOTIFY IS DISABLED -
  BACKUPS ARE MANUAL -
  CACHE IS DISABLED
ADD JOURNAL FOO_AIJ_0 -
! FILE DISK:[DIRECTORY]FOO_AIJ_0.AIJ;1
  FILE TEST$SCRATCH:FOO_AIJ_0.AIJ
ADD JOURNAL FOO_AIJ_1 -
! FILE DISK:[DIRECTORY]FOO_AIJ_1.AIJ;1
  FILE TEST$SCRATCH:FOO_AIJ_1.AIJ
ADD JOURNAL FOO_AIJ_2 -
! FILE DISK:[DIRECTORY]FOO_AIJ_2.AIJ;1
  FILE TEST$SCRATCH:FOO_AIJ_2.AIJ
ADD JOURNAL FOO_AIJ_3 -
! FILE DISK:[DIRECTORY]FOO_AIJ_3.AIJ;1
  FILE TEST$SCRATCH:FOO_AIJ_3.AIJ
ADD JOURNAL FOO_AIJ_4 -
! FILE DISK:[DIRECTORY]FOO_AIJ_4.AIJ;1
  FILE TEST$SCRATCH:FOO_AIJ_4.AIJ
ADD JOURNAL FOO_AIJ_5 -
! FILE DISK:[DIRECTORY]FOO_AIJ_5.AIJ;1
  FILE TEST$SCRATCH:FOO_AIJ_5.AIJ
ADD JOURNAL FOO_AIJ_6 -
! FILE DISK:[DIRECTORY]FOO_AIJ_6.AIJ;1
  FILE TEST$SCRATCH:FOO_AIJ_6.AIJ
ADD JOURNAL FOO_AIJ_7 -
! FILE DISK:[DIRECTORY]FOO_AIJ_7.AIJ;1
  FILE TEST$SCRATCH:FOO_AIJ_7.AIJ
$ show symbol rdm$aij*
```

Oracle® Rdb for OpenVMS

```
RDM$AIJ_BACKUP_SEQNO == "-1"
RDM$AIJ_COUNT == "8"
RDM$AIJ_CURRENT_SEQNO == "0"
RDM$AIJ_ENDOFFILE == "2"
RDM$AIJ_FULLNESS == "0"
RDM$AIJ_LAST_SEQNO == "-1"
RDM$AIJ_NEXT_SEQNO == "0"
RDM$AIJ_SEQNO == "-1"
$!
$ rmu/backup/nolog foo foo
$!
```

The second `RMU/SHOW AFTER_JOURNAL` command is executed with the `/BACKUP_CONTEXT` qualifier and the `/NOCHECKPOINT` qualifier specified. Therefore, a global checkpoint will not be executed before the global symbols that start with 'RDM\$AIJ_' are defined or modified. The `/NOCHECKPOINT` qualifier did not have to be specified since it is the default.

```
$ rmu/show after_journal/backup_context/nocheckpoint foo
JOURNAL IS ENABLED -
  RESERVE 10 -
  ALLOCATION IS 512 -
  EXTENT IS 512 -
  OVERWRITE IS DISABLED -
  SHUTDOWN_TIMEOUT IS 60 -
  NOTIFY IS DISABLED -
  BACKUPS ARE MANUAL -
  CACHE IS DISABLED
ADD JOURNAL FOO_AIJ_0 -
! FILE DISK:[DIRECTORY]FOO_AIJ_0.AIJ;1
  FILE TEST$SCRATCH:FOO_AIJ_0.AIJ
ADD JOURNAL FOO_AIJ_1 -
! FILE DISK:[DIRECTORY]FOO_AIJ_1.AIJ;1
  FILE TEST$SCRATCH:FOO_AIJ_1.AIJ
ADD JOURNAL FOO_AIJ_2 -
! FILE DISK:[DIRECTORY]FOO_AIJ_2.AIJ;1
  FILE TEST$SCRATCH:FOO_AIJ_2.AIJ
ADD JOURNAL FOO_AIJ_3 -
! FILE DISK:[DIRECTORY]FOO_AIJ_3.AIJ;1
  FILE TEST$SCRATCH:FOO_AIJ_3.AIJ
ADD JOURNAL FOO_AIJ_4 -
! FILE DISK:[DIRECTORY]FOO_AIJ_4.AIJ;1
ADD JOURNAL FOO_AIJ_5 -
! FILE DISK:[DIRECTORY]FOO_AIJ_5.AIJ;1
  FILE TEST$SCRATCH:FOO_AIJ_5.AIJ
ADD JOURNAL FOO_AIJ_6 -
! FILE DISK:[DIRECTORY]FOO_AIJ_6.AIJ;1
  FILE TEST$SCRATCH:FOO_AIJ_6.AIJ
ADD JOURNAL FOO_AIJ_7 -
! FILE DISK:[DIRECTORY]FOO_AIJ_7.AIJ;1
  FILE TEST$SCRATCH:FOO_AIJ_7.AIJ
$ show symbol rdm$aij*
RDM$AIJ_BACKUP_SEQNO == "-1"
RDM$AIJ_COUNT == "8"
RDM$AIJ_CURRENT_SEQNO == "0"
RDM$AIJ_ENDOFFILE == "2"
RDM$AIJ_FULLNESS == "0"
RDM$AIJ_LAST_SEQNO == "-1"
RDM$AIJ_NEXT_SEQNO == "0"
RDM$AIJ_SEQNO == "-1"
```

8.1.3 Engine Error Logging

This feature allows error messages returned from a database engine on a remote server to be logged. Only non success messages are logged. The server must be running Release 7.3.2.1 or higher.

These messages will typically be written into a NETSERVER.LOG file. However, they can be written to a different log file by creating an RDB\$SERVER_DEFAULTS.DAT file on the server and defining:

```
RCI_DUMP_LOGFILE "DISK:[DIR]FILE.LOG"
```

This feature is "OFF" by default. It can be turned on by the following methods.

Note: setting any of these methods to "ON" will turn the feature on. Each method can only be set to "TRUE" or "ON". All other values are ignored. Thus, if any one is set "ON" then the feature will be enabled even if another is set "OFF".

1. Define the logical RDB\$RDBSHR_ENGINEERR_LOG "ON". The logical must be set on the server so that it is visible to Dispatch. Setting it in the system table may be best. See the following example.

```
DEFINE/SYSTEM RDB$RDBSHR_ENGINEERR_LOG "ON"
```

2. Create an RDB\$SERVER_DEFAULTS.DAT file on the server and define the logical.

```
RCI_ENGINEERR_LOG "ON"
```

3. Create an RDB\$CLIENT_DEFAULTS.DAT file on the client and define the logical.

```
RCI_ENGINEERR_LOG "ON"
```

This will cause the client to instruct the server to turn on Engine Error Logging. This also requires that both client and server are running Release 7.3.2.1 or higher.

Be aware that SQL_DEFAULTS_RESTRICTION may stop RCI_ENGINEERR_LOG from being read. Thus, it is advised that RCI_ENGINEERR_LOG be in the most privileged .DAT file.

The files are read in the following order:

```
RDB$SYSTEM_DEFAULTS:RDB$SERVER_DEFAULTS.DAT
RDB$GROUP_DEFAULTS:RDB$SERVER_DEFAULTS.DAT
RDB$USER_DEFAULTS:RDB$SERVER_DEFAULTS.DAT
Then SYS$LOGIN:RDB$SERVER_DEFAULTS.DAT is read only if
RDB$USER_DEFAULTS:RDB$SERVER_DEFAULTS.DAT does not exist.
```

```
RDB$SYSTEM_DEFAULTS:RDB$CLIENT_DEFAULTS.DAT
RDB$GROUP_DEFAULTS:RDB$CLIENT_DEFAULTS.DAT
RDB$USER_DEFAULTS:RDB$CLIENT_DEFAULTS.DAT
Then SYS$LOGIN:RDB$CLIENT_DEFAULTS.DAT is read only if
RDB$USER_DEFAULTS:RDB$CLIENT_DEFAULTS.DAT does not exist.
```

If Engine Error Logging is enabled, an entry is written to the "Keyword values negotiated between client and server..." section of the log file.

This entry indicates Engine Error Logging is on:

LOGGING ENGINE ERRORS

An example of an error report:

```
** 17-MAY-2016 01:37:24.73: %RDB-E-ENGINEERR, The database engine has returned
an error for client 15a250 connection 21
%RDB-F-SYS_REQUEST, error from system services request
%RDMS-F-FILACCERR, error opening storage area file DISK1:[DATABASE]JOBS.SNP;1
***** Error while processing RCI_CLASS_REQ
```

8.1.4 New MEDIAN Aggregate Function Added to SQL

Bug 1358157

This release of Oracle Rdb includes a new statistical function, MEDIAN.

Description

MEDIAN returns the middle value of the set of ordered values for the group. If the number of values is an even number, then the result is the linear interpolation between the two middle values. NULL values are excluded from the set of values used by MEDIAN and are not counted.

MEDIAN will accept values of the following data types: TINYINT, SMALLINT, INTEGER, BIGINT, NUMERIC, DECIMAL, NUMBER, FLOAT, REAL, DOUBLE PRECISION, and INTERVAL. It returns the same data type as a result.

The following example shows the use of MEDIAN in a SQL query.

```
SQL> select employee_id,
cont>         count (*),
cont>         avg (salary_amount) as avg edit using '-(7)9.9(2)',
cont>         median (salary_amount) as median
cont> from salary_history
cont> where employee_id in ('00165', '00188')
cont> group by employee_id
cont> ;
EMPLOYEE_ID          AVG          MEDIAN
00165                12      9313.17      9017.00
00188                 2      21093.00     21093.00
2 rows selected
SQL>
```

8.1.5 New RMU/BACKUP/AFTER_JOURNAL [NO]SPACE_CHECK Qualifier

If the OpenVMS operating system detects insufficient disk space when the Oracle Rdb RMU/BACKUP/AFTER_JOURNAL command is creating, or writing to, a backup disk file or a temporary disk work file needed for the backup of one or more After Image Journal (AIJ) files, a fatal %RMU-F-FILACCERR error is output followed by the RMS-F-FUL error returned by OpenVMS and the backup operation is terminated. The point where the lack of disk space is detected can occur when a file is created by RMU/BACKUP/AFTER_JOURNAL or whenever data is being written to a

file created by RMU/BACKUP/AFTER_JOURNAL.

```
%RMU-F-FILACCERR, error creating AIJ backup file
DEVICE:[DIRECTORY]BACKUP_AIJ.AIJBCK;1
-RMS-F-FUL, device full (insufficient space for allocation)

%RMU-F-FILACCERR, error writing to backup file
DEVICE:[DIRECTORY]BACKUP_AIJ.AIJBCK;1
-RMS-F-FUL, device full (insufficient space for allocation)
```

To avoid aborting an AIJ backup because of insufficient disk space when RMU/BACKUP/AFTER_JOURNAL is creating a file or while AIJ data is being written to the created file, the [NO]SPACE_CHECK qualifier has been added to the RMU/BACKUP/AFTER_JOURNAL command to perform a disk space check at the earliest possible point in the backup: before creating the backup file for the next AIJ file to be backed up; before backing up the next AIJ file using the /RENAME optimization of creating a new version of the AIJ file; or before creating a temporary switchover file for backing up the currently active AIJ file.

The disk space check is based on the number of blocks which will be needed for the AIJ file to be backed up or renamed, or the number of blocks needed to allocate the temporary work file which will be created. The number of required blocks is compared to a snapshot of the current free blocks on the disk device where a backup file will be created, or the disk device where a new version of the backup file will be created (see the documentation for the RENAME qualifier), or the disk device where a switchover file will be created for backing up the currently active AIJ file.

The disk space check is the default and will only not be done if /NOSPACE_CHECK is specified or if the AIJ is not being backed up to a disk device. Note that this disk space check is based on a snapshot of the current disk device free space which can change immediately before or after the snapshot is taken. The %RMU-E-DISKNOSPACE message that is output if insufficient disk free space is detected specifies the free blocks available on the disk device at the time the space check is made (see below).

If there is insufficient disk space to continue the backup, the following message will be output to the operator console:

```
%RMU-I-OPERNOTIFY, system operator notification: error backing up AIJ
AFTER1 - no disk space on device DISK1:
```

The following error messages will be output:

```
%RMU-E-DISKNOSPACE, Insufficient space on device "DISK1:", needed
blocks 512, free blocks 400, total blocks 500

%RMU-F-AIJBCKNOSPACE, After journal "AFTER1" could not be backed up
because of insufficient space on device "DISK1:"
```

In the above example messages, "AFTER1" is the AIJ name in the database root, displayed as "AIJ_NAME =" by the RMU/DUMP/HEADER command, and "DISK1:" is the name of the disk device. "Needed blocks" are the required disk blocks, "free blocks" are the available free disk blocks and "total blocks" is the total of the allocated and free blocks on the disk device.

The syntax for this qualifier is:

```
/[NO]SPACE_CHECK
```

The default if this qualifier is not specified is /SPACE_CHECK.

The following example shows the /SPACE_CHECK qualifier specified with the RMU/BACKUP/AFTER command. When backing up after-image journal "AFTER1" to disk device "DISK1:", 512 blocks will be needed but only 400 blocks are free and "DISK1:" only has a maximum capacity of 500 blocks. Therefore the backup is aborted. Note that /SPACE_CHECK is the default and /NOSPACE_CHECK must be specified to bypass the space check on disk devices.

```
$ rmu/backup/after/SPACE_CHECK/continuous/until=13:53:24.33 -
  /log mf_personnel DISK:[DIRECTORY]backup_aij.out
%RMU-I-AIJBCKBEG, beginning after-image journal backup operation
%RMU-I-OPERNOTIFY, system operator notification: AIJ backup operation started
%RMU-I-AIJBCKSEQ, backing up after-image journal sequence number 0
%RMU-I-LOGBCKAIJ, backing up after-image journal AFTER1 at 13:49:24.40
%RMU-E-DISKNOSPACE, Insufficient space on device "DISK1:", needed blocks 512,
  free blocks 400, total blocks 500
%RMU-I-OPERNOTIFY, system operator notification: error backing up AIJ AFTER1 -
  no disk space on device DISK1:
%RMU-I-AIJBCKSTOP, backup of after-image journal AFTER1 did not complete
%RMU-I-OPERNOTIFY, system operator notification: AIJ manual backup operation
  failed
%RMU-F-AIJBCKNOSPACE, After journal "AFTER1" could not be backed up because
  of insufficient space on device "DISK1:"
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 2-JUN-2016 13:49:24.43
```

8.1.6 New Options to SET SQLDA Statement

Bugs 1048570, 2863911 and 9738823

The SET SQLDA Statement now supports the ENABLE and DISABLE of the FULL QUERY HEADER. This setting fills in the SQLNAME for any non-column expression in a select expression.

```
enable-option =
--> FULL QUERY HEADER -->
|
|
--> INSERT RETURNING ----+
|
|
--> INTEGER COUNT -----+
|
|
--> NAMED MARKERS -----+
|
|
--> ROWID TYPE -----+
```

The following example uses Dynamic SQL and accepts various statements. The tool displays the label from the SQLDA as a description for the user.

```
Enter statement:
attach 'filename sql$database';
inputs: 0
Enter statement:
set sqlda 'enable full query header';
inputs: 0
Enter statement:
select employee_id, first_name || last_name, extract(year from birthday)
from employees
where employee_id = '00164';
```

```

inputs: 0
out: [0] typ=Char {453} len=5
out: [1] typ=Char {453} len=24
out: [2] typ=Integer {497} len=4
[SQLDA - displaying 3 fields]
 0/EMPLOYEE_ID: 00164
 1/CONCAT(FIRST_NAME,...): Alvin      Toliver
 2/EXTRACT(YEAR FROM BIRTHDAY): 1947
Enter statement:

```

Usage Notes

- ◆ FULL QUERY HEADER – By default, any select expression that is not a column or DBKEY is given an empty SQLNAME in the SQLDA (SQLNAME_LEN is zero). When this option is enabled, an approximation of the select expression is formatted as a label for the expression. The SQLNAME_LEN will be between 1 and 62, therefore the expression may be truncated. If any of the SQLDA options ORACLE LEVEL1, ORACLE LEVEL2 or ORACLE LEVEL3 are set, then the SQLNAME_LEN will be limited to 30 as this is the largest name supported by Oracle Database. If the dialect is set to any of ORACLE LEVEL1, ORACLE LEVEL2 or ORACLE LEVEL3, then some functions will be presented using Oracle Database names (SYSDATE, SYSTIMESTAMP, ROWID and NVL) instead of the Oracle Rdb SQL names (CURRENT_TIMESTAMP, DBKEY and COALESCE) regardless of the SQL syntax used in the original query.

8.1.7 New RMU Set Statistics Command

Enhancement Bug 21618556

The Oracle Rdb RMU Set Statistics command allows the user to manage the saving and restoring of database statistics.

RMU Set Statistics allows saving database statistics by writing them to a node-specific database file located in the same directory as the database root file, and initializing database statistics by reading them from the same node-specific database file. This file has the same name as the root file, with a default file extension of .rds.

The statistics file contains node-specific information, and it cannot be renamed or copied. The exported database statistics file (.rds) can be deleted if it is no longer needed. The RMU Backup command does not save the statistics files. They are considered to be temporary files and not part of the database.

The RMU Set Statistics command can be used with databases defined with a Manual open mode and databases defined with an Automatic open mode. The RMU Close command Statistics=Export qualifier can also be used to save statistics to the same node-specific database file and the RMU Open command Statistics=Import qualifier can be used to set database statistics by reading them from the same node-specific database file, but only for databases defined with a Manual open mode. For more information, see the documentation for the RMU Close and RMU Open commands.

You must have RMU\$ALTER privilege in the access control list (ACL) for a database or the OpenVMS SYSPRV or BYPASS privilege to use the RMU Set Statistics command.

Database statistics must be enabled on the database. The RMU Dump Header command will display the following message if statistics are enabled for the database.

```
- Statistics are enabled
```

The general format for this command is:

```
RMU/Set Statistics root-file-spec
```

The qualifiers for this command are:

```
EXPORT
EXPORT = CLOSE
EXPORT = NOCLOSE
```

If just EXPORT is specified, the database monitor will immediately write the database statistics to the node-specific database statistics file. The monitor must be running and the database must currently be open. The default if EXPORT is omitted is not to write the database statistics to the node-specific database statistics file. The monitor log will record the export of the database statistics. If the node-specific database statistics file does not exist, it will be created. The existing node-specific database statistics file contents will be replaced and the version number of this file will not be incremented.

If EXPORT = CLOSE is specified, a parameter will be set in the database root to cause the monitor to automatically save the database statistics to the node-specific database statistics file whenever the database is closed. No immediate statistics export is executed. The RMU Dump Header command will show the open and close modes for the database as either:

```
Database open mode is MANUAL
Database close mode is MANUAL
```

or:

```
Database open mode is AUTOMATIC
Database close mode is AUTOMATIC
```

If the open and close modes are MANUAL, the database must be opened by an RMU Open command and closed by an RMU Close command. For more information, see the documentation for the RMU Close and RMU Open commands. If the open and close modes are AUTOMATIC, the database is opened when the first user attaches to the database and closed when no users are attached to the database.

If EXPORT = NOCLOSE is specified, a parameter will be set in the database root to cause the monitor to not save the database statistics to the node-specific database statistics file whenever the database is closed.

If EXPORT = CLOSE or EXPORT = NOCLOSE is specified, the database must currently be closed since the root parameters of the database are being modified.

If automatic exporting of the database statistics to the node-specific database statistics file is enabled for the database, the RMU Dump Header command will display the following message:

Oracle® Rdb for OpenVMS

- Statistics export on database close is enabled

If automatic exporting of the database statistics to the node-specific database statistics file is disabled for the database, the RMU Dump Header command will display the following message:

- Statistics export on database close is disabled

IMPORT = OPEN
IMPORT = NOOPEN

If IMPORT = OPEN is specified, a parameter is set in the database root to cause the monitor to automatically initialize the database statistics by reading the node-specific database statistics file whenever the database is opened. Statistics can only be imported when the database is opened. No immediate import of database statistics is allowed once the database has been opened to prevent currently active database statistics values from being overwritten.

If IMPORT = NOOPEN is specified, a parameter will be set in the database root to cause the monitor to not initialize the database statistics by reading the node-specific database statistics file when the database is opened. The RMU Dump Header command will show the open and close modes for the database as either of the following:

```
Database open mode is MANUAL  
Database close mode is MANUAL
```

or:

```
Database open mode is AUTOMATIC  
Database close mode is AUTOMATIC
```

If the open and close modes are MANUAL, the database must be opened by an RMU Open command and closed by an RMU Close command. For more information, see the documentation for the RMU Close and RMU Open commands. If the open and close modes are AUTOMATIC, the database is opened when the first user attaches to the database and closed when no users are attached to the database.

If IMPORT = OPEN or IMPORT = NOOPEN is specified, the database must currently be closed since the root parameters of the database are being modified.

If automatic importing of the database statistics from the node-specific database statistics file is enabled for the database, the RMU Dump Header command will display the following message:

- Statistics import on database open is enabled

If automatic importing of the database statistics from the node-specific database statistics file is disabled for the database, the RMU Dump Header command will display the following message:

- Statistics import on database open is disabled

CHECKPOINT
CHECKPOINT = n
NOCHECKPOINT

When statistics values are imported from the node-specific database statistics file by the database monitor at the time the database is opened, automatic periodic checkpoints are started by default to export the statistics values to the node-specific statistics file to keep the statistics file as current as possible in case a system crash occurs before the database is closed.

If only NOCHECKPOINT is specified, these default periodic export checkpoints are disabled by modifying a parameter in the database root. If only CHECKPOINT is specified, a default periodic export checkpoint interval of 30 minutes is used.

The monitor log and the RMU Show System command will indicate if these checkpoints are occurring.

All options of the checkpoint qualifier require that the database must currently be closed since the root parameters of the database are being modified.

If statistic export checkpoints are enabled for the database, the RMU Dump Header command will display the following message:

```
- Statistics export checkpoints are enabled
```

If statistic export checkpoints are disabled for the database, the RMU Dump Header command will display the following message:

```
- Statistics export checkpoints are disabled
```

By default, these periodic export checkpoints occur every 30 minutes. If CHECKPOINT = n is specified, the export checkpoints interval parameter can be modified in the database root for checkpoints to occur every n minutes, where n can be a minimum export checkpoint interval of 30 minutes, a maximum checkpoint interval of 1440 minutes (which is one checkpoint every 24 hours), or any number of minutes between these minimum and maximum values. The RMU Dump Header command will display the setting of the export checkpoint value in the root as follows:

```
- Statistics export interval is n minutes
```

If CHECKPOINT = n is specified, only the statistics export checkpoint interval for the database is modified. NOCHECKPOINT or CHECKPOINT without the checkpoint interval in minutes parameter, must be specified to disable or enable periodic export checkpoints.

If statistics are disabled for the database, the RMU Dump Header command will display the following:

```
Statistics are disabled
```

If statistics are enabled for the database but the import of database statistics on database open is disabled, the RMU Dump Header command will not show any statistics checkpoint information since statistics checkpoints are only executed if statistics are imported when the database is opened.

```
Statistics...
- Statistics are enabled
- Statistics import on database open is disabled
- Statistics export on database close is disabled
```

Oracle® Rdb for OpenVMS

If statistics are enabled for the database and the import of database statistics on database open is enabled, the RMU Dump Header command will show the statistics export checkpoint parameters for the database.

```
Statistics...
- Statistics are enabled
- Statistics import on database open is enabled
- Statistics export on database close is enabled
- Statistics export checkpoints are enabled
- Statistics export interval is 30 minutes
```

LOG
NOLOG

Displays informational messages during the execution of the RMU Set Statistics command. The default for this qualifier is NOLOG.

Examples

The following example shows the use of the RMU Set Statistics command to do an immediate export to the database node-specific statistics file to save the current database statistics.

An RMU Show System command shows that the database monitor is running and that the database is open, which are requirements for doing immediate exports and imports of statistics. No *.rds statistics file currently exists for the database so the RMU Set Statistics command Export qualifier creates the file using the node name and the database name and writes the current database statistics values to the statistics file.

Example 8-1 RMU Set Statistics Export

```
$ rmu/show user
Oracle Rdb V7.3-200 on node TSTNOD 11-AUG-2016 16:16:33.77
- monitor started 10-AUG-2016 19:14:09.13 (uptime 0 21:02:24)
- monitor log filename is "DEVICE:[DIRECTORY]RDMMON73_TSTNOD.LOG;3487"

database DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
- opened 11-AUG-2016 16:13:09.96 (elapsed 0 00:03:23)
- 2 active database users on this node

$ dir *.rds
%DIRECT-W-NOFILES, no files found
$ rmu/set statistics/export/log mf_personnel.
%RMU-I-MODIFIED, Database Statistics Import/Export file state modified
$ dir/date *.rds

Directory DEVICE:[DIRECTORY]

MF_PERSONNEL_TSTNOD.RDS;1
11-AUG-2016 16:17:05.54

Total of 1 file.
```

The following example shows the use of the RMU Set Statistics command to modify the statistics

export and import parameters in the database root used by the database monitor for the export of statistics to the database node-specific statistics file when the database is closed and the import of statistics from the database node-specific statistics file when the database is opened, as well as for periodic checkpoints to export the statistics to keep them as up to date as possible in case a system crash occurs before the database is closed.

The first RMU Show System command shows that the database is closed and not being accessed by database users, which is a requirement for setting statistics import and export parameters in the database root.

The RMU Dump Header command shows that statistics collection is enabled for the database but the export of statistics when the database is closed and the import of statistics when the database is opened are both disabled. The statistics export checkpoint parameters are not shown because statistic checkpoints are only executed if statistics are imported when the database is opened.

The RMU Set Statistics command is then used to enable statistic imports when the database is opened, statistic exports when the database is closed and to set the interval between periodic statistic exports to 60 minutes (the default is 30 minutes). Periodic export checkpoints are the default if statistics are imported when the database is opened.

The RMU Dump Header command is used to display the new statistics parameters that have been set in the database root. Later, the RMU Set Statistics command is used to disable statistic imports when the database is opened, statistic exports when the database is closed, and periodic checkpoints to export statistics.

The RMU Dump Header command shows that statistics collection is enabled for the database but the export of statistics when the database is closed and the import of statistics when the database is opened are both disabled. The statistics export checkpoint parameters are not shown because statistic checkpoints are only executed if statistics are imported when the database is opened.

Example 8–2 RMU Set Statistics Checkpoint

```
$ rmu/show system
Oracle Rdb V7.3-200 on node TSTNOD 12-AUG-2016 09:54:08.45
  - monitor started 11-AUG-2016 19:14:09.20 (uptime 0 14:39:59)
  - monitor log filename is "DEVICE:[DIRECTORY]RDMMON73_MALIBU.LOG;3488"
  - no databases are accessed by this node

$ rmu/dump/header/out=mfp.hdr mf_personnel
sear mfp.hdr statistics
Statistics...
  - Statistics are enabled
  - Statistics import on database open is disabled
  - Statistics export on database close is disabled

$ rmu/set statistics/import=open/export=close/checkpoint=60/log mf_personnel
%RMU-I-LOGMODFLG,      enabled database open statistics import
%RMU-I-LOGMODFLG,      enabled database close statistics export
%RMU-I-LOGMODFLG,      modified statistics export interval to 60 minutes
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
%RMU-I-MODIFIED, Database Statistics Import/Export file state modified

$ sear mfp.hdr statistics
Statistics...
  - Statistics are enabled
```

Oracle® Rdb for OpenVMS

- Statistics import on database open is enabled
- Statistics export on database close is enabled
- Statistics export checkpoints are enabled
- Statistics export interval is 60 minutes

```
$ rmu/set statistics/import=noopen/export=noclose/nocheckpoint/log mf_personnel
%RMU-I-LOGMODFLG,      disabled database open statistics import
%RMU-I-LOGMODFLG,      disabled database close statistics export
%RMU-I-LOGMODFLG,      disabled database statistics export checkpoints
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
%RMU-I-MODIFIED, Database Statistics Import/Export file state modified
```

```
$ sear mfp.hdr statistics
Statistics...
- Statistics are enabled
- Statistics import on database open is disabled
- Statistics export on database close is disabled
```

8.1.8 Multi-Aggregate Index Optimization

Bug 1085681

Previous versions of Oracle Rdb provided specialized optimizations to descend the index structure to compute MIN, MAX, COUNT(*), COUNT(DISTINCT expression) and COUNT(expression) aggregations.

These optimizations include:

- ◆ MAX – *Max key lookup*,
- ◆ MIN – *Min key lookup*,
- ◆ COUNT – *Index counts* (for SORTED indices), and *Index counts lookup* (for SORTED RANKED indices),
- ◆ COUNT (DISTINCT) – *Index distinct counts* (for SORTED indices), and *Index distinct lookup* (for SORTED RANKED indices).

```
SQL> select max (salary_amount)
cont> from salary_history
cont> where salary_amount is not null
cont> ;
Tables:
  0 = SALARY_HISTORY
Aggregate: 0:MAX (0.SALARY_AMOUNT) Q2
Index only retrieval of relation 0:SALARY_HISTORY
Index name SALARY_NDX [0:1] Max key lookup
Keys: NOT MISSING (0.SALARY_AMOUNT)

          93340.00
1 row selected
SQL>
```

These optimizations use specialized code to traverse the SORTED or SORTED RANKED index, which results in reduced CPU time and possibly reduced I/O to generate these results. However, these optimizations were only applied to simple queries using just one of these aggregates. For example, this meant that any query that requested both the MAX (salary_amount) and also MIN (salary_amount) was not using these optimizations for either MAX or MIN.

Oracle® Rdb for OpenVMS

```
SQL> select min (salary_amount), max (salary_amount)
cont> from salary_history
cont> where salary_amount is not null
cont> ;
Tables:
  0 = SALARY_HISTORY
Aggregate: 0:MAX (0.SALARY_AMOUNT) Q2
           1:MIN (0.SALARY_AMOUNT) Q2
Index only retrieval of relation 0:SALARY_HISTORY
  Index name  SALARY_NDX [0:1]
  Keys: NOT MISSING (0.SALARY_AMOUNT)

           7000.00          93340.00
1 row selected
SQL>
```

With this release, Oracle Rdb introduces a new method that allows the optimizer to recognize and apply many of these specialized optimizations within a single query. These aggregates can be standalone or in an expression, as shown in following example.

This query returns a single value but is now broken into three distinct index descents to efficiently compute the MAX, MIN and COUNT aggregates. In previous versions, this query would be solved by scanning the entire index and collecting the values for the computation.

```
SQL> select (MAX(salary_amount) - MIN(salary_amount))
cont>         / LEAST(1, COUNT(salary_amount))
cont>         as range_comp edit using '-(10).99'
cont> from SALARY_HISTORY;
Tables:
  0 = SALARY_HISTORY
Aggregate: 0:MAX (0.SALARY_AMOUNT) Q2
           1:MIN (0.SALARY_AMOUNT) Q2
           2:COUNT (0.SALARY_AMOUNT) Q2
Index only retrieval of relation 0:SALARY_HISTORY
  Index name  SALARY_NDX [0:0]  Max key lookup
Index only retrieval of relation 0:SALARY_HISTORY
  Index name  SALARY_NDX [0:0]  Min key lookup
Index only retrieval of relation 0:SALARY_HISTORY
  Index name  SALARY_NDX [0:1]  Index counts
  Keys: NOT MISSING (0.SALARY_AMOUNT)
  RANGE_COMP
           86340.00
1 row selected
SQL>
```

The new strategy display lists each aggregate in the order they were specified in the query.

8.1.9 Use Old DPB Format for Rdb_Change_Database

In Oracle Rdb Release 7.3.2.1, the DPB (database parameter block) parameter passed by the ALTER DATABASE statement has been augmented (extra information may be included). This may cause remote access to older versions of Oracle Rdb to fail with an RDB-F-BAD_DPB_CONTENT error. See the following example.

```
%RDB-F-BAD_DPB_CONTENT, invalid database parameters in the database parameter block (DPB)
-RDMS-E-DDLDONOTMIX, the "SYNONYMS ARE ENABLED" clause can not be used with some ALTER
DATABASE clauses
```

To use the old format when using the ALTER DATABASE statement on an older Oracle Rdb database, define the keyword RCI_OLD_CHANGE_DATABASE "ON" in RDB\$CLIENT_DEFAULTS.DAT.

8.1.10 LogMiner State Now in AIJ Options File, New RDM\$LOGMINER_STATE Symbol

Bug 23076123

The current state of LogMiner operations on an Oracle Rdb database, whether LogMiner is enabled or disabled and whether the Continuous LogMiner feature is enabled, will now be displayed either by the RMU SHOW AFTER_JOURNAL command or the new RDM\$LOGMINER_STATE symbol. The state can now be set in the AIJ options file read by the AIJ_OPTIONS qualifier used with the RMU COPY_DATABASE, RMU MOVE_AREA, RMU RESTORE, RMU RESTORE ONLY_ROOT and RMU SET AFTER_JOURNAL commands to define the database after-image journal configuration. The RMU SHOW AFTER_JOURNAL OUTPUT qualifier now includes the current LogMiner state defined in the database root in the AIJ options file. The RMU SHOW AFTER_JOURNAL BACKUP_CONTEXT qualifier now defines the new RDM\$LOGMINER_STATE string symbol.

The new LogMiner syntax added to the AIJ options file and displayed by the RMU SHOW AFTER_JOURNAL command is the following.

```
LOGMINER (IS) ENABLED|DISABLED [CONTINUOUS]
```

– "IS" is optional and does not have to be specified.

```
LOGMINER IS DISABLED
```

– Both LogMiner and the Continuous LogMiner feature are disabled. The Continuous LogMiner feature cannot be enabled unless LogMiner is also enabled.

```
LOGMINER IS ENABLED
```

– LogMiner is enabled but the Continuous LogMiner feature is disabled.

```
LOGMINER IS ENABLED CONTINUOUS
```

– Both LogMiner and the Continuous LogMiner feature are enabled. The Continuous LogMiner feature cannot be enabled unless LogMiner is also enabled.

The new RDM\$LOGMINER_STATE string symbol can have the following values.

```
$ SHOW SYMBOL RDM$LOGMINER_STATE
RDM$LOGMINER_STATE == "DISABLED"
```

– Both LogMiner and the Continuous LogMiner feature are disabled. The Continuous LogMiner feature cannot be enabled unless LogMiner is also enabled.

```
$ SHOW SYMBOL RDM$LOGMINER_STATE
RDM$LOGMINER_STATE == "ENABLED"
```


Oracle® Rdb for OpenVMS

– LogMiner is enabled but the Continuous LogMiner feature is disabled.

```
$ SHOW SYMBOL RDM$LOGMINER_STATE
RDM$LOGMINER_STATE == "ENABLED CONTINUOUS"
```

– Both LogMiner and the Continuous LogMiner feature are enabled. The Continuous LogMiner feature cannot be enabled unless LogMiner is also enabled.

In the following example, the RMU SHOW AFTER_JOURNAL command shows that LogMiner and the Continuous LogMiner feature are disabled for a database and defines the RDM\$LOGMINER_STATE symbol to indicate this. The database is then backed up with LogMiner disabled. Then journaling is enabled with one variable size extensible AIJ file and the RMU SET LOGMINER command is used to enable LogMiner for the database but keeps the Continuous LogMiner feature disabled. The Continuous LogMiner feature requires a fixed size file circular AIJ configuration. The RMU SHOW AFTER_JOURNAL command is then used to create an AIJ options file with LogMiner enabled and the Continuous LogMiner feature disabled and a variable size file extensible AIJ configuration. Then the database is deleted and restored from the backup file with both journaling and LogMiner disabled, but because the RMU RESTORE AIJ_OPTIONS qualifier is specified, the RMU RESTORE command reads the AIJ options file created by the RMU SHOW AFTER_JOURNAL command and restores the variable size file extensible AIJ configuration with LogMiner enabled and the Continuous LogMiner feature disabled.

```
$ RMU/SHOW AFTER_JOURNAL/BACKUP_CONTEXT-
/OUT=SHOW.TMP DEVICE:[DIRECTORY]MF_PERSONNEL
$ SEAR SHOW.TMP LOGMINER
LOGMINER IS DISABLED
$ SHOW SYMBOL RDM$LOGMINER_STATE
RDM$LOGMINER_STATE == "DISABLED"
$ RMU/BACKUP/NOLOG DEVICE:[DIRECTORY]MF_PERSONNEL -
DEVICE:[DIRECTORY]MFP
$ SQL
ALTER DATABASE FILE DEVICE:[DIRECTORY]MF_PERSONNEL
JOURNAL IS ENABLED
ADD JOURNAL TEST1 FILENAME 'TEST1.AIJ';
%RDMS-W-DOFULLBCK, full database backup should be done to ensure
future recovery
EXIT;
$ RMU/SET LOGMINER/ENABLE/NOCONTINUOUS/LOG -
DEVICE:[DIRECTORY]MF_PERSONNEL
%RMU-I-MODIFIED, LogMiner state modified
%RMU-W-DOFULLBCK, full database backup should be done to ensure
future recovery
$ RMU/SHOW AFTER_JOURNAL/BACKUP_CONTEXT/OUT=AIJ1OPT.OPT -
DEVICE:[DIRECTORY]MF_PERSONNEL
$ TYPE AIJ1OPT.OPT
JOURNAL IS ENABLED -
RESERVE 1 -
ALLOCATION IS 512 -
EXTENT IS 512 -
OVERWRITE IS DISABLED -
SHUTDOWN_TIMEOUT IS 60 -
NOTIFY IS DISABLED -
BACKUPS ARE MANUAL -
CACHE IS DISABLED -
LOGMINER IS ENABLED
ADD JOURNAL TEST1 -
! FILE DISK:[DIRECTORY]TEST1.AIJ;1
FILE TEST1.AIJ
```

Oracle® Rdb for OpenVMS

```
$ SHOW SYMBOL RDM$LOGMINER_STATE
RDM$LOGMINER_STATE == "ENABLED"
$!
$ SQL
  DROP DATABASE FILE DEVICE:[DIRECTORY]MF_PERSONNEL;
  EXIT
$ DELETE *.AIJ;*
$ RMU/RESTORE/NOCCD/NOLOG/DIR=TEST$SCRATCH-
/AIJ_OPTIONS=DEVICE:[DIRECTORY]AIJ1OPT.OPT -
DEVICE:[DIRECTORY]MFP
  JOURNAL IS ENABLED -
  RESERVE 1 -
  ALLOCATION IS 512 -
  EXTENT IS 512 -
  OVERWRITE IS DISABLED -
  SHUTDOWN_TIMEOUT IS 60 -
  NOTIFY IS DISABLED -
  BACKUPS ARE MANUAL -
  CACHE IS DISABLED -
  LOGMINER IS ENABLED
ADD JOURNAL TEST1 -
! FILE DISK:[DIRECTORY]TEST1.AIJ;1
  FILE TEST1.AIJ
$ RMU/SHOW AFTER_JOURNAL/BACKUP_CONTEXT/OUT=SHOW.TMP -
DEVICE:[DIRECTORY]MF_PERSONNEL
$ SEAR SHOW.TMP LOGMINER
  LOGMINER IS ENABLED
$ SHOW SYMBOL RDM$LOGMINER_STATE
RDM$LOGMINER_STATE == "ENABLED"
```

In the following example, the RMU SHOW AFTER_JOURNAL command shows that LogMiner and the Continuous LogMiner feature are disabled for a database and defines the RDM\$LOGMINER_STATE symbol to indicate this. The database is then backed up with LogMiner disabled. Then journaling is enabled with a fixed size file circular AIJ file configuration and the RMU SET LOGMINER command is used to enable both LogMiner and the Continuous LogMiner option for the database. The Continuous LogMiner feature requires a fixed size file circular AIJ configuration. The RMU SHOW AFTER_JOURNAL command is then used to create an AIJ options file with both LogMiner and the Continuous LogMiner option enabled with a fixed size file circular AIJ configuration. Then the database is deleted and restored from the backup file with both journaling and LogMiner disabled. An RMU SET AFTER_JOURNAL command is then used with the AIJ_OPTIONS qualifier to read the AIJ options file created by the RMU SHOW AFTER_JOURNAL command to restore the fixed size file AIJ configuration with both LogMiner and the Continuous LogMiner feature enabled.

```
$ RMU/SHOW AFTER_JOURNAL/BACKUP_CONTEXT/OUT=SHOW.TMP -
DEVICE:[DIRECTORY]MF_PERSONNEL
$ SEAR SHOW.TMP LOGMINER
  LOGMINER IS DISABLED
$ SHOW SYMBOL RDM$LOGMINER_STATE
RDM$LOGMINER_STATE == "DISABLED"
$ RMU/BACKUP/NOLOG DEVICE:[DIRECTORY]MF_PERSONNEL -
DEVICE:[DIRECTORY]MFP
$ SQL
ALTER DATABASE FILE DEVICE:[DIRECTORY]MF_PERSONNEL
  JOURNAL IS ENABLED
  RESERVE 5 JOURNALS
  ADD JOURNAL TEST1 FILENAME 'TEST1.AIJ'
  ADD JOURNAL TEST2 FILENAME 'TEST2.AIJ';
%RDMS-W-DOFULLBCK, full database backup should be done
```

Oracle® Rdb for OpenVMS

```
to ensure future recovery
%RDMS-W-DOFULLBCK, full database backup should be done
to ensure future recovery
EXIT;
$ RMU/SET LOGMINER/ENABLE/CONTINUOUS/LOG -
  DEVICE:[DIRECTORY]MF_PERSONNEL
%RMU-I-MODIFIED, LogMiner state modified
%RMU-W-DOFULLBCK, full database backup should be done to
  ensure future recovery
$ RMU/SHOW AFTER_JOURNAL/BACKUP_CONTEXT/OUT=AIJ1OPT.OPT -
  DEVICE:[DIRECTORY]MF_PERSONNEL
$ TYPE AIJ1OPT.OPT
JOURNAL IS ENABLED -
  RESERVE 6 -
  ALLOCATION IS 512 -
  EXTENT IS 512 -
  OVERWRITE IS DISABLED -
  SHUTDOWN_TIMEOUT IS 60 -
  NOTIFY IS DISABLED -
  BACKUPS ARE MANUAL -
  CACHE IS DISABLED -
  LOGMINER IS ENABLED CONTINUOUS
ADD JOURNAL TEST1 -
! FILE DISK:[DIRECTORY]TEST1.AIJ;1
  FILE TEST1.AIJ
ADD JOURNAL TEST2 -
! FILE DISK:[DIRECTORY]TEST2.AIJ;1
  FILE TEST2.AIJ
$ SHOW SYMBOL RDM$LOGMINER_STATE
  RDM$LOGMINER_STATE == "ENABLED CONTINUOUS"
$!
$ SQL
  DROP DATABASE FILE DEVICE:[DIRECTORY]MF_PERSONNEL;
  EXIT
$ DELETE *.AIJ;*
$ RMU/RESTORE/NOCCD/NOLOG/DIR=TEST$SCRATCH -
  DEVICE:[DIRECTORY]MFP
%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.
$ RMU/SHOW AFTER_JOURNAL/BACKUP_CONTEXT/OUT=SHOW.TMP -
  DEVICE:[DIRECTORY]MF_PERSONNEL
$ SEAR SHOW.TMP LOGMINER
  LOGMINER IS DISABLED
$ SHOW SYMBOL RDM$LOGMINER_STATE
  RDM$LOGMINER_STATE == "DISABLED"
$ RMU/SET AFTER_JOURNAL/AIJ_OPTIONS=DEVICE:[DIRECTORY]AIJ1OPT.OPT -
  DEVICE:[DIRECTORY]MF_PERSONNEL
%RMU-I-RESTXT_18, Processing options file
DEVICE:[DIRECTORY]AIJ1OPT.OPT
  JOURNAL IS ENABLED -
    RESERVE 6 -
    ALLOCATION IS 512 -
    EXTENT IS 512 -
    OVERWRITE IS DISABLED -
    SHUTDOWN_TIMEOUT IS 60 -
    NOTIFY IS DISABLED -
    BACKUPS ARE MANUAL -
    CACHE IS DISABLED -
    LOGMINER IS ENABLED CONTINUOUS
  ADD JOURNAL TEST1 -
```

Oracle® Rdb for OpenVMS

```
! FILE DISK:[DIRECTORY]TEST1.AIJ;1
  FILE TEST1.AIJ
ADD JOURNAL TEST2 -
! FILE DISK:[DIRECTORY]TEST2.AIJ;1
  FILE TEST2.AIJ
%RMU-I-LOGMODFLG,      disabled after-image journaling
%RMU-I-LOGMODVAL,      reserved 6 additional after-image journals
%RMU-W-DOFULLBCK, full database backup should be done to ensure
  future recovery
%RMU-I-LOGMODFLG,      enabled LogMiner
%RMU-I-LOGMODFLG,      enabled LogMiner
%RMU-I-LOGMODFLG,      disabled after-image journal file overwrite
%RMU-I-LOGMODVAL,      modified AIJ shutdown time to 60 minutes
%RMU-I-LOGMODFLG,      disabled after-image journal spooler
%RMU-I-LOGMODVAL,      modified after-image journal file
  allocation to 512
%RMU-I-LOGMODVAL,      modified after-image journal file
  extension to 512
%RMU-I-LOGMODSTR,      switching from extensible to circular
  AIJ journaling
%RMU-I-LOGCREAIJ, created after-image journal file
  DISK:[DIRECTORY]TEST1.AIJ;1
%RMU-I-LOGMODSTR,      added after-image journal definition "TEST1"
%RMU-I-LOGCREAIJ, created after-image journal file
  DISK:[DIRECTORY]TEST2.AIJ;1
%RMU-I-LOGMODSTR,      added after-image journal definition "TEST2"
%RMU-I-LOGMODSTR,      activated after-image journal "TEST1"
%RMU-I-LOGMODFLG,      enabled after-image journaling
%RMU-W-DOFULLBCK, full database backup should be done to ensure
  future recovery
$ RMU/SHOW AFTER_JOURNAL/BACKUP_CONTEXT/OUT=SHOW.TMP -
  DEVICE:[DIRECTORY]MF_PERSONNEL
$ SEAR SHOW.TMP LOGMINER
  LOGMINER IS ENABLED CONTINUOUS
$ SHOW SYMBOL RDM$LOGMINER_STATE
  RDM$LOGMINER_STATE == "ENABLED CONTINUOUS"
```

The following example shows that, although the LogMiner Continuous LogMiner feature requires a fixed size file circular AIJ configuration to execute, the RMU SET LOGMINER command allows the LogMiner Continuous LogMiner feature to be enabled with a variable size file extensible AIJ configurations but outputs the warning message:

```
%RMU-W-CLMCIRCAIJ, Continuous LogMiner requires fixed-size circular
after-image journals
```

This indicates that the database must be altered to have a fixed size file circular AIJ configuration or the Continuous LogMiner feature will fail with the following fatal error:

```
%RMU-F-CLMCIRCAIJ, Continuous LogMiner requires fixed-size circular
after-image journals
```

The RMU SHOW AFTER_JOURNAL command now will also issue the following warning message if it detects that the LogMiner Continuous LogMiner feature is enabled with a variable size file extensible AIJ configuration.

```
%RMU-W-CLMCIRCAIJ, Continuous LogMiner requires fixed-size circular
after-image journals
```

Here is an example of this situation.

```

$ SQL
ALTER DATABASE FILE DEVICE:[DIRECTORY]MF_PERSONNEL
  JOURNAL IS ENABLED
  ADD JOURNAL TEST1 FILENAME 'TEST1.AIJ';
%RDMS-W-DOFULLBCK, full database backup should be done to
ensure future recovery
EXIT;
$ RMU/SHOW AFTER_JOURNAL/BACKUP_CONTEXT DEVICE:[DIRECTORY]MF_PERSONNEL
JOURNAL IS ENABLED -
  RESERVE 1 -
  ALLOCATION IS 512 -
  EXTENT IS 512 -
  OVERWRITE IS DISABLED -
  SHUTDOWN_TIMEOUT IS 60 -
  NOTIFY IS DISABLED -
  BACKUPS ARE MANUAL -
  CACHE IS DISABLED -
  LOGMINER IS DISABLED
ADD JOURNAL TEST1 -
! FILE DISK:[DIRECTORY]TEST1.AIJ;1
  FILE TEST1.AIJ
$ SHOW SYMBOL RDM$LOGMINER_STATE
  RDM$LOGMINER_STATE == "DISABLED"
$ RMU/SET LOGMINER/ENABLE/CONTINUOUS/LOG DEVICE:[DIRECTORY]MF_PERSONNEL
%RMU-W-CLMCIRCAIJ, Continuous LogMiner requires fixed-size circular
after-image journals
%RMU-I-MODIFIED, LogMiner state modified
%RMU-W-DOFULLBCK, full database backup should be done to ensure
future recovery
$ RMU/SHOW AFTER_JOURNAL/BACKUP_CONTEXT/OUT=AIJ1OPT.OPT -
  DEVICE:[DIRECTORY]MF_PERSONNEL
%RMU-W-CLMCIRCAIJ, Continuous LogMiner requires fixed-size circular
after-image journals
$ TYPE AIJ1OPT.OPT
JOURNAL IS ENABLED -
  RESERVE 1 -
  ALLOCATION IS 512 -
  EXTENT IS 512 -
  OVERWRITE IS DISABLED -
  SHUTDOWN_TIMEOUT IS 60 -
  NOTIFY IS DISABLED -
  BACKUPS ARE MANUAL -
  CACHE IS DISABLED -
  LOGMINER IS ENABLED CONTINUOUS
ADD JOURNAL TEST1 -
! FILE DISK:[DIRECTORY]TEST1.AIJ;1
  FILE TEST1.AIJ
$ SHOW SYMBOL RDM$LOGMINER_STATE
  RDM$LOGMINER_STATE == "ENABLED CONTINUOUS"

```

8.1.11 New /[NO]MBX_ASYNC Qualifier for RMU/UNLOAD/AFTER

Bug 24514893

In a prior version of Oracle Rdb, an enhancement was added to the RMU/UNLOAD/AFTER command which affected how unloaded records were written when the output device was a VMS mailbox.

Previously, when the RMU Logminer (LM) utility wrote a record to the mailbox, it would wait for another process to read that record from the mailbox before posting another record. In Oracle Rdb Release 7.3.1.3, the write operation was changed to be more asynchronous in order to increase throughput. The LM process no longer had to wait for the reader operation to finish before writing more unloaded records. The enhancement was always enabled and could not be disabled.

It has been discovered that in some situations, it is possible for the LM to saturate the I/O buffer causing the entire process to hang, which could cause the AIJ Backup Server (ABS) to stall and block other database users. This would typically occur when an after-image journal switchover was attempted. Searching RMU/SHOW STAT stall logs would show many processes stalled with "waiting for AIJ journal control" messages. The only workaround to free the stalled processes was to delete the LM process(es).

Starting with release 7.3.2.1 of Oracle Rdb, a new qualifier, /[NO]MBX_ASYNCH, has been added to the RMU/UNLOAD/AFTER command to control the behavior of how the LM writes records to the mailbox.

The qualifier /MBX_ASYNCH is enabled by default. It provides the asynchronous mailbox write mechanism. If you experience users stalled with "waiting for AIJ journal control" while running LM, you can restart the LM with the /NOMBX_ASYNCH qualifier to revert to the old synchronous mailbox write behavior.

8.1.12 New /PAGE_NUMBER Qualifier for RMU/DUMP and RMU/DUMP/BACKUP

Release 7.3.2.1 of Oracle Rdb adds a new qualifier to RMU/DUMP and RMU/DUMP/BACKUP. The /PAGE_NUMBER qualifier is a shortcut for the case where both /END and /START specify a single page number. Neither /END nor /START can be specified if /PAGE_NUMBER is used.

8.1.13 New Information Table RDB\$SESSION_PRIVILEGES Now Available

Bug 5300887

This release of Oracle Rdb includes a new information table, RDB\$SESSION_PRIVILEGES, that returns a decoding (or mapping) of the database access control privileges similar to Oracle Database system privileges. While there is not a precise one-to-one mapping between Rdb privileges and Oracle system privileges, this is adequate to allow many Oracle OCI applications to successfully query the data dictionary view, SESSION_PRIVS, based upon this information table.

This table includes a single text column, RDB\$PRIVILEGE, which contains a string describing an assigned privilege.

```
SQL> select rdb$privilege from rdb$session_privileges;  
RDB$PRIVILEGE
```

Oracle® Rdb for OpenVMS

```
CREATE SESSION
1 row selected
SQL>
```

A revised version of the script `SQL$SAMPLE:INFO_TABLES.SQL` is provided by this release. It can be executed to drop and recreate the information tables.

The following example shows the execution of this script to add the new information table.

```
SQL> @SQL$SAMPLE:INFO_TABLES.SQL

Copyright (c) 1997, 2016, Oracle Corporation. All Rights Reserved.

Please ignore any FIELD_EXISTS error for the domain RDB$FILE_SPECIFICATION

Creating RDB$CACHES

Creating RDB$DATABASE_JOURNAL

Creating RDB$DATABASE_ROOT

Creating RDB$DATABASE_USERS

Creating RDB$STORAGE_AREAS

Creating RDB$JOURNALS

Creating RDB$LOGICAL_AREAS

Creating RDB$CHARACTER_SETS

Creating RDB$NLS_CHARACTER_SETS

Creating RDB$SESSION_PRIVILEGES

Type COMMIT if there were no unexpected errors, otherwise ROLLBACK
SQL> commit;
SQL>
```

Chapter 9

Enhancements And Changes Provided in Oracle Rdb Release 7.3.2.0

9.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.2.0

9.1.1 New COMPRESSION OCTETS Clause for CREATE INDEX Statement

This release of Oracle Rdb allows user specification of the octets used for run length compression.

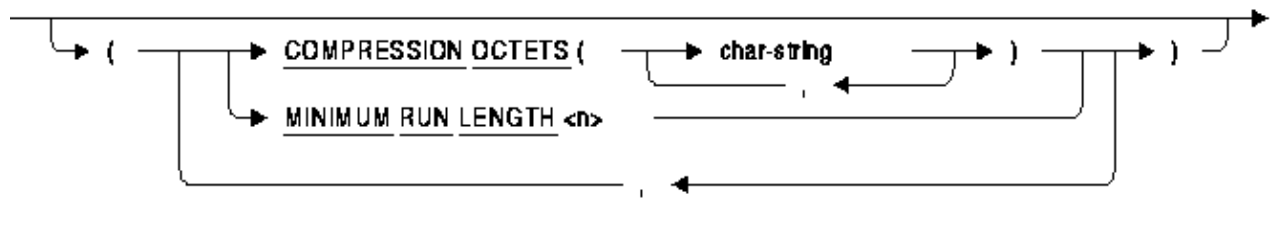
The COMPRESSION OCTETS clause can be used to specify those characters that will be used for run-length compression. If this clause is not defined, then Oracle Rdb will use the space character of the character set of any CHAR, or VARCHAR columns in the index, and a zero for any other data type.

The following example shows the use of *compression octets* clause. Here the database administrator noticed that -1 was used as a flag in the table and since many rows use the DEFAULT value, it makes sense to include x'FF' as a compression character.

```
SQL> create table PEOPLE
cont>   (name      char (100) default ' '
...
cont>   ,use_indicator      bigint default -1
cont>   )
cont> ;
SQL>
SQL> create index PEOPLE_NDX3
cont>   on PEOPLE (use_indicator)
cont>   enable compression
cont>   (minimum run length 2,
cont>   compression octets (X'00', X'FF'))
cont>   store in PEOPLE_NDX3_AREA
cont> ;
SQL>
SQL> insert into PEOPLE
cont>   default values;
1 row inserted
SQL>
```

Syntax

rlc-attr =



Usage Notes

- ◆ When *compression octets* is specified, the character strings can be simple strings, such as '*-!', or if the characters are non-printing then use the hex string specification, such as X'00A1'.

9.1.2 Enhancements for TRUNCATE TABLE Statement

This release of Oracle Rdb supports additional clauses for the TRUNCATE TABLE statement as specified by the ANSI and ISO Database Language Standard.

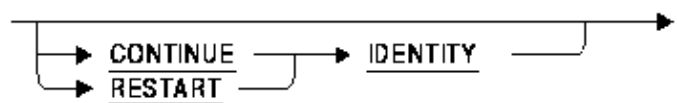
By default, the TRUNCATE TABLE statement restarts the associated IDENTITY column using the START WITH value of the sequence. There are now two new clauses to control the action of TRUNCATE TABLE when an IDENTITY column exists. Either the RESTART IDENTITY action or CONTINUE IDENTITY actions can be specified. The default behavior, when neither clause is used, and when DIALECT 'SQL2011' is used is to CONTINUE IDENTITY. For all other dialects, the default is RESTART IDENTITY (which is maintained for backward compatibility).

The following example requests that the IDENTITY column not be restarted after the truncate.

```
SQL> truncate table HISTORY_LOG continue identity;
```

Syntax

TRUNCATE TABLE <table-name>

*Arguments*

- ◆ table-name
Specifies the name of the table you want to truncate. This name must be a base table or global temporary table. Views and location temporary tables may not be truncated.
- ◆ CONTINUE IDENTITY
Requests that TRUNCATE TABLE statement leave the current next value unchanged for the associated IDENTITY column.
- ◆ RESTART IDENTITY
Requests that TRUNCATE TABLE reset the associated IDENTITY column so that it starts with the START WITH value or, if there is none, the MINVALUE value defined for the sequence.

Usage Notes

- ◆ You must have DELETE privilege for the table, as this command deletes all data.
- ◆ You must have CREATE privilege at the table level.
- ◆ If there exists an AFTER DELETE or BEFORE DELETE trigger defined on this table, you will require DROP and CREATE privileges for triggers on this table. These privileges are required because this operation effectively disables these triggers.
- ◆ TRUNCATE TABLE is a data definition statement and as such requires exclusive access to the table.

- ◆ The TRUNCATE TABLE statement fails with an error message if:
 - ◇ RDB\$SYSTEM storage area is set to read-only
 - ◇ The named table is a view
 - ◇ The named table has been reserved for data definition
 - ◇ The named table is a system table
- ◆ TRUNCATE TABLE deletes all data in the table, however, it does not execute any BEFORE or AFTER DELETE triggers.
- ◆ If the dialect is set to SQL2011 and neither CONTINUE IDENTITY nor RESTART IDENTITY clauses are specified, the default will be CONTINUE IDENTITY. For all other dialects, the default is RESTART IDENTITY.
- ◆ TRUNCATE TABLE explicitly resets the values in Rdb\$WORKLOAD rows associated with this table, as well as removing any index or table storage statistics.
- ◆ All CHECK and FOREIGN KEY constraints that reference the truncated table are revalidated after the truncate operation to ensure that the database remains consistent. If constraint validation fails, the TRUNCATE statement is automatically rolled back. For example:

```
SQL> set dialect 'sql99';
SQL> CREATE TABLE test1
cont> (col1 REAL PRIMARY KEY);
SQL> CREATE TABLE test2
cont> (col1 REAL REFERENCES TEST1 (COL1));
SQL> INSERT INTO test1 VALUES (1);
1 row inserted
SQL> INSERT INTO test2 VALUES (1);
1 row inserted
SQL> COMMIT;
SQL>
SQL> TRUNCATE TABLE test1;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-INTEG_FAIL, violation of constraint TEST2_FOREIGN1 caused operation to fail
-RDB-F-ON_DB, on database USERS2:[TESTING.DATABASES.]PERSONNEL.RDB;1
SQL> TABLE test1;
          COL1
          1.0000000E+00
1 row selected
SQL> ROLLBACK;
```

- ◆ When a table contains one or more LIST OF BYTE VARYING columns, the TRUNCATE TABLE statement must read each row in the table and record the pointers for all LIST values. This list is processed at COMMIT time to delete the LIST column data. Therefore, the database administrator must also allow for this time when truncating the table. Reserving the table for EXCLUSIVE WRITE is recommended because the dropped LIST columns will require that each row in the table be updated and set to NULL – it is this action which queues the pointers for commit time processing. This reserving mode will eliminate snapshot file I/O, lower lock resources and reduce virtual memory usage. As the LIST data is stored outside the table, performance may be improved by attaching to the database with the RESTRICTED ACCESS clause, which has the side effect of reserving all the LIST storage areas for EXCLUSIVE access and therefore eliminates snapshot I/O during the delete of the LIST data.
- ◆ If table contains no LIST OF BYTE VARYING columns, and the table and all associated indices are stored in UNIFORM storage areas, then TRUNCATE TABLE will employ the most efficient mechanism to erase the data from the table.

9.1.3 RMU Extract Now Supports RECOMPILE Item

In this release of Oracle Rdb, RMU Extract supports a new Item keyword: RECOMPILE. This keyword causes RMU to generate a series of ALTER MODULE commands to recompile any invalid routines. The default is to only include those modules that have at least one invalid routine. Use Option=FULL to generate a script that compiles all modules.

The following example shows the generated SQL script for a database with routines that were marked invalid. RMU annotates the generated ALTER MODULE statement with the names of those invalid routines.

```
$ rmu/extract/option=(noheader,filename_only) abc /item=recompile
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename EXAMPLE';

alter module JACKET compile;
-- Invalid Procedure GU
-- Invalid Procedure GBS
-- Invalid Function F_GBS_DIRECT
commit work;
$
```

9.1.4 SQL Now Supports the MISSING VALUE Clause as Part of CREATE and ALTER DOMAIN Statement

Bug 19882369

This release of Oracle Rdb adds a MISSING VALUES FOR RDO clause to SQL. This functionality has been supported by the Rdb Server for users of the RDO interface since the earliest releases of Rdb.

This new clause is provided in SQL to allow applications written using the RDBPRE precompilers and databases created using the RDO database definition language to be migrated to SQL-only interfaces and still retain this functionality.

Note

Oracle does not recommend the use of MISSING VALUE for modern applications because these semantics are not defined by the ANSI nor ISO SQL Database Language Standards. MISSING VALUE was introduced for RDO as a way to manage NULL (unknown) values in the database. SQL supports well defined standard functions IS NULL, NULLIF, and COALESCE for such purposes, as well as allowing applications to fetch the null indication into an INDICATOR variable.

The following example shows the specification of MISSING VALUE on a CREATE DOMAIN

statement.

```
SQL> create domain STATUS_CODE
cont>     CHAR (1)
cont>     check((value in ('0', '1', '2')
cont>         or (value is null)))
cont>     not deferrable
cont>     missing value for RDO is 'N'
cont>     comment is
cont>         ' A number';
SQL>
```

The CREATE and ALTER DOMAIN statements allow the MISSING VALUE clause to be defined for a domain. The ALTER DOMAIN statement allows an existing MISSING VALUE clause to be removed using the NO MISSING VALUE clause. All tables that use these domains will implicitly inherit the new semantics.

The literal value specified for the MISSING VALUE must be compatible with the data type of the domain. Only simple literal expressions are supported; namely DATE VMS literals, character string literals and numeric values (fixed or floating).

The RMU/EXTRACT/ITEM=DOMAIN command will implicitly output the new MISSING VALUE FOR RDO clause for any domain that has that attribute defined. Previously, a log message would report that it was not supported and only /LANGUAGE=RDO would extract the MISSING VALUE clause.

9.1.5 Comma Statement Separator Now Deprecated

The syntax for trigger actions in the CREATE TRIGGER statement has supported the comma (,) as well as the semicolon (;) as statement separators. The use of the comma separator has been problematic in Oracle Rdb SQL because it conflicts in various places with the comma used as an element separator within some statements. For example, the TRACE statement allows a comma separated list of values, and the INSERT INTO ... SELECT ... FROM statement allows a comma separated list of table names in the FROM clause. In these cases, a comma can not be used as a statement separator because the current statement appears to be continued.

Future versions of Oracle Rdb are expected to include enhancements to the TRIGGER action syntax which will allow other statements to include comma as an element separator. Therefore, the comma statement separator is now deprecated. A future functionality release of Oracle Rdb will remove or severely restrict the usage of the comma separator. Therefore, Oracle recommends that any scripts or applications that include the CREATE TRIGGER statement be modified to use only the semicolon (;) as a separator.

The following example shows the new diagnostic issued by Interactive SQL. Similar diagnostics are issued by the SQL Module Language compiler and the SQL Precompiler.

```
SQL> create trigger employee_id_cascade_insert
cont>     after insert on employees
cont>         (insert into job_history (employee_id) values (employees.employee_id)
cont>         ,
%SQL-I-DEPR_FEATURE, Deprecated Feature: use ; instead of , as a statement
separator in trigger actions
cont>         insert into salary_history (employee_id) values (employees.employee_id)
```

```

cont>      ,
%SQL-I-DEPR_FEATURE, Deprecated Feature: use ; instead of , as a statement
separator in trigger actions
cont>      insert into degrees (employee_id) values (employees.employee_id)
cont>      )
cont>      for each row;
SQL>

```

This change does not affect existing database triggers, only new triggers defined using the CREATE TRIGGERS statement. RMU Extract Item=TRIGGERS command already uses the semicolon separator in extracted CREATE TRIGGER statements.

9.1.6 New Logical Name RDMS\$BIND_DEADLOCK_WAIT to Control Sub-second Deadlock Wait

OpenVMS allows the system manager to establish a DEADLOCK_WAIT by setting this system parameter as small as 1 second. OpenVMS will establish a default value of 10 seconds. More recent releases of OpenVMS allow applications to establish a process specific DEADLOCK_WAIT smaller than 1 second using the system service SYS\$SET_PROCESS_PROPERTIESW.

This release of Oracle Rdb provides an interface to this system service for Rdb applications that wish to make use of sub-second DEADLOCK_WAIT times. The logical name RDMS\$BIND_DEADLOCK_WAIT can be defined to a numeric value that specifies the deadlock wait time in 100 nanosecond units. The smallest value is 100000 (which is 10 milliseconds) and the largest value is 10000000 (which is 1 second). If the value specified for the logical name is outside this range, it will be ignored and the application will default to the setting of the system parameter DEADLOCK_WAIT.

This logical name can be defined as a process, group, job or system wide logical name. Rdb only translates this logical name on the first database connection. Any effects of this logical name are removed when the image which attached to the database exits.

Please note that the smaller the deadlock wait setting, the more often the OpenVMS lock manager will initiate a deadlock search. The use of the logical name for Rdb is only recommended for high-end transaction processing systems which have the database load and sufficiently powerful CPU systems to require such fine tuning.

9.1.7 Query Optimization Improvements for IN Clause

Bugs 12548885 and 14471918

The EXISTS and IN predicates can often be used interchangeably in queries to check for the existence of values in another result set. If possible, the EXISTS query should be the first preference because its structure allows for the best query optimization. However, the semantics of these predicates are not identical when NULL values are present in one or both tables, especially when used with the NOT operator. Care should be taken to ensure correct query behavior in such cases.

With this release of Oracle Rdb, the optimizer will attempt to transform the IN predicate to an EXISTS predicate when the source columns are known to be not nullable. Such a transformation will

Oracle® Rdb for OpenVMS

return the same results and additionally present a better query for optimization.

The following example shows the strategy selected for NOT IN when the optimization is not (or cannot be) applied.

```
SQL> select s.badge_number
cont> from STAFF s
cont> where s.badge_number NOT IN (select kb.badge_number from KNOWN_BADGES kb)
cont> ;
Tables:
  0 = STAFF
  1 = KNOWN_BADGES
Cross block of 2 entries  Q1
  Cross block entry 1
    Index only retrieval of relation 0:STAFF
      Index name  STAFF_I [0:0]
  Cross block entry 2
    Conjunct: <agg0> = 0
    Aggregate-F1: 0:COUNT-ANY (<subselect>) Q2
    Conjunct: MISSING (0.BADGE_NUMBER) OR MISSING (1.BADGE_NUMBER) OR (
              0.BADGE_NUMBER = 1.BADGE_NUMBER)
    Index only retrieval of relation 1:KNOWN_BADGES
      Index name  KNOWN_BADGES_I [0:0]
  BADGE_NUMBER
      4
1 row selected
SQL>
```

When the target columns (for example BADGE_NUMBER) in each table have a NOT DEFERRABLE constraint of the type PRIMARY KEY or NOT NULL, then the following strategy is used. The resulting strategy will likely result in faster query execution.

```
SQL> select s.badge_number
cont> from STAFF s
cont> where s.badge_number NOT IN (select kb.badge_number from KNOWN_BADGES kb)
cont> ;
Tables:
  0 = STAFF
  1 = KNOWN_BADGES
Conjunct: <agg0> = 0
Match      (Agg Outer Join)  Q1
  Outer loop
    Match_Key:0.BADGE_NUMBER
      Index only retrieval of relation 0:STAFF
        Index name  STAFF_I [0:0]
  Inner loop      (zig-zag)
    Match_Key:1.BADGE_NUMBER
    Index_Key:BADGE_NUMBER
      Aggregate-F1: 0:COUNT-ANY (<subselect>) Q2
      Index only retrieval of relation 1:KNOWN_BADGES
        Index name  KNOWN_BADGES_I [0:0]
  BADGE_NUMBER
      4
1 row selected
SQL>
```

This transformation is enabled by default but can be disabled using SET FLAGS 'NOWRITE(IN_CLAUSE)' and re-enabled using SET FLAGS 'REWRITE(IN_CLAUSE)'.

This new feature was actually introduced in Oracle Rdb Release 7.3.1 but was inadvertently left out of the release notes.

9.1.8 New SHOW AUDIT Command Added to Interactive SQL

This release of Oracle Rdb adds a SHOW AUDIT command to Interactive SQL. Modeled closely on the SHOW PROTECTION and SHOW PRIVILEGES commands, it allows the database administrator to display audit and alarm information for each database object.

When using the database ALIAS, information is displayed about the database such as whether auditing is enabled or not and the list of identifiers (users and roles) and privileges that trigger auditing. For other database objects (tables, views, sequences, modules, procedures, functions and columns), only the privileges for audit and alarms are displayed (if any).

The following example shows the output for SHOW AUDIT ON DATABASE. A list of two database ALIAS are specified, one database has auditing enabled, the other has no auditing.

```
SQL> SHOW AUDIT ON DATABASE RDB$DBHANDLE, DB1;
Audit information for Alias RDB$DBHANDLE
Auditing is enabled
Alarms will be written to the operator
Audit every object access
Forced writes of audit journal records is enabled
Audit Event Classes:
    PROTECTION (Grant and Revoke)
    DACCESS (Discretionary Access)
Identifiers:
    [DEV,TEST_EXECUTE]
    [PRD,*]
    [MGR,ADMIN]
    [AUD,*]
Audit Privileges:
    SELECT,DISTRIBTRAN
Alarm Privileges:
    DROP,SECURITY

Audit information for Alias DB1
Auditing is disabled

SQL>
```

9.1.9 RMU/RECLAIM Can Now Skip to the Next SPAM Interval and/or Storage Area to Avoid Lock Contention

Bug 11813831

The Oracle Rdb RMU/RECLAIM command reclaims deleted dbkeys and locked space from database pages. It is designed for customer sites where database users attach to an Oracle Rdb database in DBKEY SCOPE IS ATTACH mode.

To avoid lock contention with other database users, RMU/RECLAIM sequentially fetches data pages in a storage area but does not process data pages in the storage area that are currently locked in an incompatible mode by other users. However, once RMU/RECLAIM is processing a page, it can conflict with other database users who must wait for the page that RMU/RECLAIM is processing. RMU/RECLAIM is designed to run in the background and avoid lock contention with other users; this new optional feature will detect that other users are currently locking pages in the same storage area or SPAM interval that RMU/RECLAIM is processing and skip to the next storage area or SPAM interval based on a percent of the pages that RMU/RECLAIM is not able to process in a SPAM interval or storage area when reaching a specified limit.

For this new functionality, the following new qualifiers have been added to the RMU/RECLAIM command.

◆ /PAGE_SKIP_LIMIT[=n]

If this qualifier is specified without the qualifier /SPAM_SKIP_LIMIT, the current storage area will be skipped by RMU/RECLAIM once this percent of pages in the area have been ignored by RMU/RECLAIM due to lock contention with other users. If this qualifier is specified with the qualifier /SPAM_SKIP_LIMIT, the current SPAM interval will be abandoned and RMU will skip to the next SPAM interval in the current storage area once this percent of pages in the current SPAM interval have been ignored by RMU/RECLAIM due to lock contention with other users. The minimum value that can be specified is 1 percent. The maximum value that can be specified is 100 percent. The default value is 25 percent.

◆ /SPAM_SKIP_LIMIT[=n]

This qualifier specifies the percent of SPAM intervals that can be skipped in the current storage area due to lock contention with other users before the current storage area is skipped and the next storage area is processed. This qualifier can only be specified if the /PAGE_SKIP_LIMIT qualifier is also specified. The minimum value that can be specified is 1 percent. The maximum value that can be specified is 100 percent. The default value is 25 percent.

◆ /RETRY_TIME[=n]

Before completing, RMU/RECLAIM will perform one retry of the processing of any storage areas that have been skipped when /PAGE_SKIP_LIMIT or /SPAM_SKIP_LIMIT have been specified. The /RETRY_TIME qualifier specifies a wait time in seconds before RMU/RECLAIM reprocesses the storage areas where pages and SPAM intervals were previously skipped in order to allow more time for page contention to decrease. This qualifier can only be specified if the /PAGE_SKIP_LIMIT qualifier is also specified. The minimum value that can be specified is 0 seconds. The maximum value that can be specified is 3600 seconds (one hour). The default value is 60 seconds.

In the following examples, the /LOG qualifier is specified to show the new messages that will be output if the /LOG qualifier is specified. The first example shows that the %RMU-I-RCLMPRCT message will be output even if this new feature is not used to show the percentage of pages that the RMU/RECLAIM command was able to process and did not have to skip because other users were accessing those pages in an incompatible locking mode. The second example shows just a /PAGE_SKIP_LIMIT of 1 percent being specified causing the DEPARTMENTS storage area to be skipped. The immediate reprocessing of the area has the same result. In the third example, the immediate reprocessing of the area succeeds. In the fourth example, a /RETRY_TIMEOUT wait of 20 seconds is specified before the reprocessing of the area, which has the same results. In the fifth example, a /SPAM_SKIP_LIMIT of 1 percent is also specified. A /RETRY_TIMEOUT wait of 60 seconds is specified and the reprocessing of the area succeeds.

```
$ RMU/RECLAIM/LOG/AREA=DEPARTMENTS MF_PERSONNEL
```

Oracle® Rdb for OpenVMS

```
%RMU-I-RCLMAREA, Reclaiming area DEPARTMENTS
%RMU-I-RCLMPRCT, 704 pages processed of 706 total pages for area DEPARTMENTS,
approximately 99 %
  ELAPSED:    0 00:00:00.18  CPU: 0:00:00.02  BUFIO: 20  DIRIO: 257  FAULTS: 207
$
$ RMU/RECLAIM/LOG/AREA=SALARY_HISTORY/PAGE_SKIP_LIMIT=1 MF_PERSONNEL
%RMU-I-RCLMAREA, Reclaiming area SALARY_HISTORY
%RMU-I-RCLMPRCT, 1 pages processed of 706 total pages for area SALARY_HISTORY,
approximately 1 %
%RMU-I-RCLMRTRY, Retrying processing of area DEV:[DIR]SALARY_HISTORY.RDA;1
that did not complete
%RMU-I-RCLMAREA, Reclaiming area DEV:[DIR]SALARY_HISTORY.RDA;1
%RMU-I-RCLMPRCT, 1 pages processed of 706 total pages for area
DEV:[DIR]SALARY_HISTORY.RDA;1, approximately 1 %
%RMU-I-RCLMNOPRC, Retry of area DEV:[DIR]SALARY_HISTORY.RDA;1 did not complete
due to continued lock contention
  ELAPSED:    0 00:01:00.11  CPU: 0:00:00.04  BUFIO: 29  DIRIO: 23  FAULTS: 201
$
$ RMU/RECLAIM/LOG/AREA=SALARY_HISTORY/PAGE_SKIP_LIMIT=1 MF_PERSONNEL
%RMU-I-RCLMAREA, Reclaiming area SALARY_HISTORY
%RMU-I-RCLMPRCT, 1 pages processed of 706 total pages for area SALARY_HISTORY,
approximately 1 %
%RMU-I-RCLMRTRY, Retrying processing of area DEV:[DIR]SALARY_HISTORY.RDA;1
that did not complete
%RMU-I-RCLMAREA, Reclaiming area DEV:[DIR]SALARY_HISTORY.RDA;1
%RMU-I-RCLMPRCT, 706 pages processed of 706 total pages for area
DEV:[DIR]SALARY_HISTORY.RDA;1, approximately 100 %
%RMU-I-RCLMPROC, Retry of area DEV:[DIR]SALARY_HISTORY.RDA;1 succeeded
  ELAPSED:    0 00:01:00.11  CPU: 0:00:00.04  BUFIO: 29  DIRIO: 23  FAULTS: 201
$
$ RMU/RECLAIM/LOG/AREA=SALARY_HISTORY/PAGE_SKIP_LIMIT=1/RETRY_TIMEOUT=20
MF_PERSONNEL
%RMU-I-RCLMAREA, Reclaiming area SALARY_HISTORY
%RMU-I-RCLMPRCT, 1 pages processed of 706 total pages for area SALARY_HISTORY,
approximately 1 %
%RMU-I-RCLMWAIT, Reclaim waiting for 20 seconds before reprocessing areas that
did not complete
%RMU-I-RCLMRTRY, Retrying processing of area DEV:[DIR]SALARY_HISTORY.RDA;1
that did not complete
%RMU-I-RCLMAREA, Reclaiming area DEV:[DIR]SALARY_HISTORY.RDA;1
%RMU-I-RCLMPRCT, 1 pages processed of 706 total pages for area
DEV:[DIR]SALARY_HISTORY.RDA;1, approximately 1 %
%RMU-I-RCLMNOPRC, Retry of area DEV:[DIR]SALARY_HISTORY.RDA;1 did not complete
due to continued lock contention
  ELAPSED:    0 00:00:01.11  CPU: 0:00:00.02  BUFIO: 29  DIRIO: 27  FAULTS: 200
$
$ RMU/RECLAIM/LOG/AREA=DEPARTMENTS/PAGE_SKIP_LIMIT=1/SPAM_SKIP_LIMIT=1
/RETRY_TIMEOUT=60 MF_PERSONNEL
%RMU-I-RCLMAREA, Reclaiming area DEPARTMENTS
%RMU-I-RCLMPRCT, 1 pages processed of 706 total pages for area DEPARTMENTS,
approximately 1 %
%RMU-I-RCLMWAIT, Reclaim waiting for 60 seconds before reprocessing areas that
did not complete
%RMU-I-RCLMRTRY, Retrying processing of area DEV:[DIR]DEPARTMENTS.RDA;1 that
did not complete
%RMU-I-RCLMAREA, Reclaiming area DEV:[DIR]DEPARTMENTS.RDA;1
%RMU-I-RCLMPRCT, 706 pages processed of 706 total pages for area
DEV:[DIR]DEPARTMENTS.RDA;1, approximately 100 %
%RMU-I-RCLMPROC, Retry of area DEV:[DIR]DEPARTMENTS.RDA;1 succeeded
  ELAPSED:    0 00:00:01.10  CPU: 0:00:00.02  BUFIO: 29  DIRIO: 23  FAULTS: 200
$
```

9.1.10 RMU Open Statistics Supports PROCESS_GLOBAL Qualifier

The RMU Open command, with the Statistics qualifier, now supports the optional keyword [No]Process_Global, which indicates whether or not you are able to collect per-process statistics. The Statistics=Process_Global qualifier indicates that all processes attached to the database are eligible for per-process monitoring. The default qualifier, Statistics=Noprocess_Global, indicates that attached processes are not automatically eligible; however, you can still activate these processes at run-time using one of the other activation methods described below.

When you use the global activation method, the RMU Show Statistics utility changes the screen header Mode attribute from Online to Global.

Oracle Corporation recommends this method only when there are a small number of active processes. The Statistics=Process_Global qualifier causes each process attached to the database on that node to create a sizable global section into which the process global statistic collection occurs. Oracle Corporation recommends that you activate process global statistic collection on a per-process basis. That is, you should activate only those processes you are currently interested in, and deactivate them when you are finished with them.

Refer to the RMU Show Statistics Handbook for further details on per-process monitoring with RMU Show Statistics.

This example shows the PROCESS_GLOBAL option as well as an indication from RMU/SHOW USER that it has been specified.

Example 9-1 Example showing use of PROCESS_GLOBAL option

```

$ RMU/OPEN/STATISTICS=(IMPORT,PROCESS_GLOBAL) MF_PERSONNEL
$ RMU/SHOW USER MF_PERSONNEL
Oracle Rdb V7.3-13 on node GANDLF  4-MAR-2015 08:23:26.93
  - monitor started  3-MAR-2015 19:14:07.75 (uptime 0 13:09:19)
  - monitor log filename is "$1$DGA231:[LOGS]RDMMON731_GANDLF.LOG;2991"

Database $1$DGA170:[DATABASES.V73]MF_PERSONNEL.RDB;1
  - opened  4-MAR-2015 08:22:52.63 (elapsed 0 00:00:34)
  * database is opened by an operator
  * statistic information import failed
  * global per-process statistic collection activated
  * next statistic information checkpoint at  4-MAR-2015 08:52:52.66

```

9.1.11 RMU/SHOW LOGICAL_NAME Now Supports /DESCRIPTION Qualifier

Bugs 3264793, 3682207, 5634563, and 19545970

With this release of Oracle Rdb, the RMU Show Logical_Name command includes a Description qualifier. This new qualifier retrieves a brief description of the logical name and displays it along with the current definition. If wildcards are used for the logical name, then any matching logical names

will also include output of the description.

The following example shows the use of the Description qualifier, a wildcard logical name specification and the use of the Undefined qualifier to include output – even for logical names not defined for this process.

```
$ RMU/SHOW LOGICAL_NAME/UNDEFINE/DESCRIPTION RDMS$BIND_WORK*
"RDMS$BIND_WORK_FILE" = Undefined
```

You can define this logical name to redirect the location of temporary files that Oracle Rdb creates for use in matching operations. These temporary files are deleted when they are no longer used.

See also the logical name RDMS\$BIND_WORK_VM.

```
"RDMS$BIND_WORK_VM" = Undefined
```

This logical name permits you to reduce the overhead of disk I/O for matching operations by letting you specify the amount of virtual memory (VM), in bytes, to be allocated to your process. Once the allocation is exhausted, additional data values will be written to a temporary file on disk.

If the logical name RDMS\$BIND_WORK_FILE is undefined, the temporary file is located in SYS\$LOGIN, otherwise the value defined by RDMS\$BIND_WORK_FILE will be used.

The default is 100,000 bytes. The maximum allowed value is restricted only by the amount of memory available on your system.

The definitions of all logical names are maintained in a HELP library called SYS\$HELP:RMUDISPLAY73.HLB. Users can also use the DCL HELP command to query this help library.

```
$ HELP/LIBR=SYS$HELP:RMUDISPLAY73.HLB Rdb_Logical_names RDMS$RUJ
```

```
RDB_LOGICAL_NAMES
```

```
RDMS$RUJ
```

You can use the RDMS\$RUJ logical name to locate the .ruj file on a different disk and directory from the default directory. This can help to reduce contention in that directory.

Topic?

9.1.12 Using Per–Process Monitoring for RMU Show Statistics

This feature has actually been available since the early Rdb 7.1 releases but the documentation about it seems to have gotten lost. Therefore, we are including it here again.

RMU Show Statistics has a Per–Process Monitoring facility to provide a powerful drill–down

capability to allow a database administrator to analyze process-specific information for a single process, class of processes, or all attached database processes. The facility also provides several screens that display a side-by-side comparison of individual process statistic information such as I/O, transaction, and record statistics.

The Per-Process Monitoring facility presents real-time information and, consequently, does not write its information to the binary output file. Therefore, the Per-Process Monitoring facility is not available during the replay of a binary input file.

The Per-Process Monitoring facility is also not available if cluster wide statistic collection is active. Conversely, the cluster wide statistic collection facility is not available when the Per-Process Monitoring facility is active.

The RMU Show Statistics utility requires SYSGBL privilege in order to use the Per-Process Monitoring facility. This privilege is required even if the facility activation is implicitly performed by the database monitor. See [Section 9.1.12.2, Per-Process Monitoring Facility Activation](#) for activation details.

9.1.12.1 Per-Process Monitoring Operational Modes

The RMU Show Statistics utility provides two separate operational modes for the Per-Process Monitoring facility. These are:

- ◆ Process Overview Information
This operational mode allows you to compare statistic information for various activated processes for the purpose of easily identifying performance and behavioral discrepancies. The information presented is very similar to the main-menu summary screen information.
- ◆ Process Drill-Down Information
This operational mode allows you to operate the RMU Show Statistics utility against a specific attached process. All of the statistic screens display information for that process only.

Together, these operational modes allow you to completely analyze process information, especially as it correlates to the statistic information of other processes.

9.1.12.2 Per-Process Monitoring Facility Activation

You can select the Per-Process Monitoring facility using any of the following methods:

- ◆ Global Activation
Global per-process statistic collection means that all processes attached to the database are automatically eligible to be monitored using the RMU Show Statistics utility. The RMU Open command, with the Statistics qualifier, now supports the optional keyword [No]Process_Global, which indicates whether or not you are able to collect per-process statistics. The Statistics=Process_Global qualifier indicates that all processes attached to the database are eligible for per-process monitoring. The default qualifier, Statistics=Noprocess_Global, indicates that attached processes are not automatically eligible; however, you can still activate these processes at run-time using one of the other activation methods described below.

When you use the global activation method, the RMU Show Statistics utility changes the screen header Mode attribute from Online to Global.

Oracle Corporation recommends this method only when there are a small number of active

processes. The `Statistics=Process_Global` qualifier causes each process attached to the database on that node to create a sizable global section into which the process global statistic collection occurs. Oracle Corporation recommends that you activate process global statistic collection on a per-process basis. That is, you should activate only those processes you are currently interested in, and deactivate them when you are finished with them. The following formula (expressed in bytes) defines the global section memory requirements for each process when it is activated:

$$4096 + (512 * \text{"\# storage areas"}) + ((128 * \text{"\# row caches"}) / 4) + (512 * \text{"max \# logical areas"})$$

The maximum number of logical areas is determined using the `RMU Dump Header` command. Search the generated output for the line containing "Logical area count is n". Note that the logical area count is not necessarily the actual number of tables and indexes contained in the database.

For example, the `MF_PERSONNEL.RDB` database requires approximately 300K (580 pages) of global section backing store memory for each attached process by the facility. Obviously, this memory requirement increases as the size of the database increases.

◆ **Local Activation**

Local per-process statistic collection means that selected processes attached to the database are automatically eligible to be monitored from the `RMU Show Statistics` utility.

You can define the `RDM$BIND_GLOBAL_STATISTICS` logical name, located in the `LN$FILE_DEV` name table, to identify those non-server database processes that are accessible. The default value "0" indicates that the process is not eligible for monitoring. The value "1" indicates that the process is eligible for monitoring.

You can designate the database server processes eligibility using server-specific logical names (see below) also located in the `LN$SYSTEM_TABLE` name table.

Note that the database recovery process (DBR) does not participate in the per-process monitoring facility.

Oracle Corporation recommends this method only when you know in advance which processes, or class of processes, you intend to monitor. However, this is seldom the case.

◆ **Dynamic Activation**

You can dynamically activate processes attached to the database, at run-time, for per-process monitoring using the `RMU Show Statistics` utility. You commonly use this method when you want to monitor a few specific processes that exhibit run-time behavior that requires more investigation.

Refer to [Section 9.1.12.4, Per-Process Monitoring Run-Time Options](#) for more information on this using this method.

The following table summarizes the logical names and their respective name table:

Logical Name	Name Table	Affected Process
<code>RDM\$BIND_GLOBAL_STATISTICS</code>	<code>LN\$FILE_DEV</code>	Application processes
<code>RDM\$BIND_ABS_GLOBAL_STATISTICS</code>	<code>LN\$SYSTEM_TABLE</code>	AIJ Backup Server
<code>RDM\$BIND_ALS_GLOBAL_STATISTICS</code>	<code>LN\$SYSTEM_TABLE</code>	AIJ Log Server
<code>RDM\$BIND_LCS_GLOBAL_STATISTICS</code>	<code>LN\$SYSTEM_TABLE</code>	Log Catch-up Server
<code>RDM\$BIND_LRS_GLOBAL_STATISTICS</code>	<code>LN\$SYSTEM_TABLE</code>	Log Replication Server
<code>RDM\$BIND_RCS_GLOBAL_STATISTICS</code>	<code>LN\$SYSTEM_TABLE</code>	Row Cache Server

9.1.12.3 Per-Process Monitoring Facility Process Activation

When you activate the Per-Process Monitoring facility, attached database processes are in one of the following states:

- ◆ **Inactive State**
By default, attached database processes are inactive for per-process monitoring. This means that their per-process statistic information is not globally available to the RMU Show Statistics utility.
You can activate inactive processes at run-time. Refer to [Section 9.1.12.4, Per-Process Monitoring Run-Time Options](#) for more information.
- ◆ **Eligible State**
Processes that are eligible for per-process monitoring, but that you are not currently monitoring, are identified on RMU Show Statistics utility screens with the "G" suffix in the Process.ID field. The "G" suffix means statistics are being collected globally for this process, but cannot be displayed by the RMU Show Statistics utility.
- ◆ **Activated State**
The RMU Show Statistics utility is able to display overview information for any eligible process. The RMU Show Statistics utility identifies any process activated for per-process monitoring with the "A" suffix in the Process.ID field.
The fundamental difference between eligible and activated processes is that the RMU Show Statistics utility displays process-specific information for activated processes. You are only able to activate eligible processes.
For example, if you open the database using the RMU Open Statistics=Process_Global command, the "G" suffix identifies all eligible processes, as shown in the following screen:

```
Node: GANDLF (1/3/24)   Oracle Rdb V7.3-13 Perf. Monitor 16-JUN-2014 15:47:28.89
Rate: 1.00 Second           Active User Stall Messages   Elapsed: 159 01:35:39.64
Page: 1 of 1                USER_TEST:[DB_TESTING.TCS_MASTER]TCS.RDB;3           Mode: Global
-----
Process.ID  Elapsed.... T Stall.reason..... Lock.ID.
2A51334D:1s          waiting for AIJ journal lock 0 (PW)           3E007869
2A50D94E:1s          waiting for standby database activity request
2A54A759:53 02:40:05.77 W waiting for page 1:1091 (PR)                 5D00FE93
2A51A75A:1G 02:58:00.85 W waiting for page 3:2181 (PR)                 1D005613
2A55155B:1G 02:40:04.45 W waiting for page 1:1091 (PW)                 7000495D
2A53135C:1G 02:47:42.57 W waiting for page 1:1200 (PW)                 1D0087FB
2A51C603:1G 02:40:09.20 W waiting for page 1:1091 (PW)                 7500B779
2A559A06:1G 02:47:36.84 W waiting for page 33:1711 (PW)                0700EDD2
2A50FA1A:1G 02:56:11.84 R waiting for page 1:1006 (PR)                 5500EC46
2A469E1F:1G 02:47:05.52 W waiting for record 50:9591:0 (PR)                4600FC19
2A49061C:1G          locking page 33:1713
-----
Config Exit Help LockID Menu >next_page <prev_page PageInfo Set_rate Write Zoom
```

When the RMU Show Statistics utility activates eligible processes, the "A" suffix identifies these processes, as shown in the following screen:

```
Node: GANDLF (1/3/24)   Oracle Rdb V7.3-13 Perf. Monitor 16-JUN-2014 15:47:28.89
Rate: 1.00 Second           Active User Stall Messages   Elapsed: 159 01:35:39.64
```

Oracle® Rdb for OpenVMS

```

Page: 1 of 1          USER_TEST:[DB_TESTING.TCS_MASTER]TCS.RDB;3          Mode: Global
-----
Process.ID  Elapsed.... T Stall.reason..... Lock.ID.
2A51334D:1s          waiting for AIJ journal lock 0 (PW)          3E007869
2A50D94E:1s          waiting for standby database activity request
2A54A759:53 02:40:05.77 W waiting for page 1:1091 (PR)          5D00FE93
2A51A75A:1G 02:58:00.85 W waiting for page 3:2181 (PR)          1D005613
2A55155B:1G 02:40:04.45 W waiting for page 1:1091 (PW)          7000495D
2A53135C:1G 02:47:42.57 W waiting for page 1:1200 (PW)          1D0087FB
2A51C603:1A 02:40:09.20 W waiting for page 1:1091 (PW)          7500B779
2A559A06:1A 02:47:36.84 W waiting for page 33:1711 (PW)          0700EDD2
2A50FA1A:1G 02:56:11.84 R waiting for page 1:1006 (PR)          5500EC46
2A469E1F:1G 02:47:05.52 W waiting for record 50:9591:0 (PR)          4600FC19
2A49061C:1G          locking page 33:1713
-----
Config Exit Help LockID Menu >next_page <prev_page PageInfo Set_rate Write Zoom

```

Note that it is possible to activate specific processes and leave others as eligible for future activation.

Upon startup, the RMU Show Statistics utility automatically activates any eligible processes. You can either implicitly or explicitly activate eligible processes that attach to the database after the RMU Show Statistics utility, depending on which screen you are currently displaying.

The following table summarizes the possible Process.ID suffix values:

Type Tag	Description
(blank)	Ordinary application process.
D	Process is being recovered by a database recovery (DBR) process.
R	Process is a remote-connection server.
s	Process is a database server (note that the "s" suffix is lowercase to differentiate it from the character "8").
u	Process is a database utility (note that the "u" suffix is lowercase to differentiate it from the character "0").
*	Process is attached on a remote node.
A	Process is activated by the RMU Show Statistics utility for global statistic collection.
G	Process is eligible for global statistic collection but is not activated.

9.1.12.4 Per-Process Monitoring Run-Time Options

The RMU Show Statistics utility provides run-time options to manage the Per-Process Monitoring facility. The run-time options are available in the Tools menu, obtained using the exclamation mark (!). Selecting the Process Monitoring option displays one or more of the following menu options:

- ◆ Activate all eligible processes
 The RMU Show Statistics utility displays this option when the current screen is not one of the process overview screens. Process overview screens automatically activate all eligible processes.
 Selecting this option activates all eligible processes on the current node.

- ◆ De-activate all processes
The RMU Show Statistics utility displays this option when the current screen is not one of the process overview screens. Process overview screens automatically activate all eligible processes.
Selecting this option de-activates all previously activated processes. However, they are still eligible to be re-activated if needed.
- ◆ Activate specific eligible process
The RMU Show Statistics utility displays this option when the current screen is not one of the process overview screens. Process overview screens automatically activate all eligible processes.
Selecting this option activates the selected eligible processes on the current node that are not currently activated.
Note that you cannot activate database server processes using this option. You must explicitly define the RDM\$BIND_XXX_GLOBAL_STATISTICS logical name to activate database server processes.
Also, it may take a few seconds for the selected process to activate itself. Activating an already activated process has no effect on the selected process.
- ◆ De-activate specific process
The RMU Show Statistics utility displays this option when the current screen is not one of the process overview screens. Process overview screens automatically activate all eligible processes.
Selecting this option de-activates the selected, previously activated processes.
- ◆ Select activated process to monitor
The RMU Show Statistics utility displays this option when there are one or more processes activated for per-process monitoring.
Selecting this option allows the RMU Show Statistics utility screens to display information for the specified process only.
The Node portion of the screen header displays the identifier of the selected process you are currently monitoring.
- ◆ Cancel process monitoring
The RMU Show Statistics utility displays this option when you have previously selected a specific activated process to monitor.
Selecting this option reverts the RMU Show Statistics utility screens to display information for all database processes.

9.1.12.5 Detached Process Monitoring

If the process actively being monitored detaches from the database, the RMU Show Statistics utility screens will inquire whether or not you wish to continue monitoring the information of the detached process.

If you wish to continue monitoring the detached process, the process identifier in the screen header will blink. The blinking process identifier lets you know that the process information you are viewing is stale. In the event of an abnormal process termination, reviewing the stale information can be useful for determining the cause of the termination.

If you do not wish to continue monitoring the detached process, the RMU Show Statistics utility will automatically revert to displaying information for all database processes.

There is no method available to activate a process once it has detached from the database.

9.1.12.6 Per-Process Monitoring Overview Information

The RMU Show Statistics utility contains seven new screens, located in the Process Information menu, to provide overview and side-by-side comparison of information for all attached processes. The screens provide information similar to the main menu summary screens.

All of the process overview screens implicitly attach to any eligible process not already attached.

Note that you are unable to transfer statistic information from these screens onto the Custom Statistics screen.

The following is a list of the new screens:

- ◆ Process IO Overview
- ◆ Process Journal IO Overview
- ◆ Process Lock Overview
- ◆ Process Object Overview
- ◆ Process Transaction Overview
- ◆ Process Record Overview
- ◆ Process Snapshot Overview

Process IO Overview Screen

The Process IO Overview screen summarizes database I/O activity for each attached process.

```

Node: GANDLF (1/3/24)   Oracle Rdb V7.3-13 Perf. Monitor 17-JUN-2014 14:44:29.66
Rate: 1.00 Second           Process IO Overview           Elapsed: 160 00:32:40.41
Page: 1 of 1                USER_TEST:[DB_TESTING.TCS_MASTER]TCS.RDB;2           Mode: Global
-----
Process.ID Sync.Reads SyncWrites Read.Stall WriteStall AsyncReads AsyncWrits
2A46BE1F:1s          0          0          0          0          0          0
2A562A21:1s          0          0          0          0          0          0
2A56002D:1A         1058         4221         8230         11400         0          0
2A53DC2E:1A          566          560          333          9729         0          0
2A540E31:1A         5948         6764         32374         8377         0          0
2A532E37:1A          543          453          689          9240         0          0
2A52A43B:1A          0            0            0            0            0          0
2A536E3A:1A          198          1634          743          3660         13         0
2A50C047:1A          0            0            0            0            0          0
2A4CDE49:1A          0            0            0            0            0          0
-----
Exit Help Menu >next_page <prev_page Options Set_rate Write Zoom !

```

The screen fields are the following:

- ◆ Process.ID
Identifies the attached database process. All processes on this screen contain the "A" suffix unless the process is a database server.
- ◆ Sync.Reads
This field gives the number of synchronous read queued I/O requests (QIO) issued to the database storage area for single-file and multi-file databases (.RDA) and snapshot (.SNP) files. This operation reads database pages synchronously from the database.

- ◆ SyncWrites
This field gives the number of synchronous write queued I/O requests (QIO) issued to the database storage area for single-file and multi-file databases (.RDA) and snapshot (.SNP) files. This operation writes modified database pages synchronously back to the database.
- ◆ Read.Stall
This field gives the time in hundredths of a second spent reading database pages from the database rootfile (.RDB), storage area (.RDA) files, and snapshot (.SNP) files. An excessively high number often indicates disk contention that might be alleviated by moving some files to other disks.
This statistic field includes both synchronous and asynchronous I/O read stall durations.
- ◆ WriteStall
This field gives the time in hundredths of a second spent writing database pages to the database rootfile (.RDB), storage area (.RDA), and snapshot (.SNP) files. An excessively high number often indicates disk contention that might be alleviated by moving some files to other disks.
This statistic field includes both synchronous and asynchronous I/O write stall durations.
- ◆ AsyncReads
This field gives the number of asynchronous read queued I/O requests (QIO) issued to the database storage area for single-file and multi-file databases (.RDA) and snapshot (.SNP) files. This operation reads database pages asynchronously from the database.
- ◆ AsyncWrites
This field gives the number of asynchronous write queued I/O requests (QIO) issued to the database storage area for single-file and multi-file databases (.RDA) and snapshot (.SNP) files. This operation writes modified database pages asynchronously back to the database.

Process Journal IO Overview Screen

The Process Journal IO Screen summarizes journal I/O activity for each attached process.

```
Node: GANDLF (1/3/24)   Oracle Rdb V7.3-13 Perf. Monitor 17-JUN-2014 14:44:31.20
Rate: 1.00 Second      Process Journal IO Overview   Elapsed: 160 00:32:41.95
Page: 1 of 1          USER_TEST:[DB_TESTING.TCS_MASTER]TCS.RDB;2      Mode: Global
-----
Process.ID RUJ.Reads.  RUJ.Writes  RUJ.Extend  AIJ.Reads.  AIJ.Writes  AIJ.Extend
2A46BE1F:1s          0            0            0           41          3960         0
2A562A21:1s          0            0            0          197           1           0
2A56002D:1A          0            71           1            4            0           0
2A53DC2E:1A          0           152           1            4            0           0
2A540E31:1A          0            13           1            3            0           0
2A532E37:1A          0           117           1            3            0           0
2A52A43B:1A          0            0            0            0            0           0
2A536E3A:1A          0            2            1            2            0           0
2A50C047:1A          0            0            0            0            0           0
2A4CDE49:1A          0            0            0            0            0           0
-----
Exit Help Menu >next_page <prev_page Options Set_rate Write Zoom !
```

The screen fields are the following:

- ◆ Process.ID
Identifies the attached database process. All processes on this screen contain the "A" suffix unless the process is a database server.

- ◆ **RUJ.Reads**
This field gives the number of read queued I/O requests (QIO) issued to the database recovery unit journal (.RUJ) files. This operation reads before–image records from the .RUJ file to roll back a verb or a transaction.
This statistic field includes both synchronous and asynchronous I/O read requests.
- ◆ **RUJ.Writes**
This field gives the number of write queued I/O requests (QIO) issued to the database recovery unit journal (.RUJ) files. This operation writes before–image records to the .RUJ file in case a verb or transaction must be rolled back. Before–images must be written to the .RUJ file before the corresponding database page can be written back to the database.
This statistic field includes both synchronous and asynchronous I/O write requests.
- ◆ **RUJ.Extend**
This field identifies the number of times an RUJ file has been extended. Ideally, this value should be "0" or as close to "0" as possible. Each .RUJ file extension represents a performance bottleneck that is easily resolved.
- ◆ **AIJ.Reads.**
The number of read queued I/O requests (QIO) issued to the database after–image journal (.AIJ) file. If after–image journaling is not enabled for the database, this statistic will be zero.
This statistic field includes both synchronous and asynchronous I/O read requests.
- ◆ **AIJ.Writes**
This field gives the total number of write queued I/O requests (QIO) issued to the database after–image journal (.AIJ) file. If after–image journaling is not enabled for the database, this statistic will be zero. This operation writes after–image records to the after–image journal to facilitate roll–forward recovery using the RMU Recover command.
This statistic field includes both synchronous and asynchronous I/O write requests.
- ◆ **AIJ.Extend**
This field identifies the number of times an AIJ journal has been extended. Ideally, this value should be "0" or as close to "0" as possible. Each AIJ journal extension represents a performance bottleneck that is easily resolved.

Process Lock Overview Screen

The Process Lock Overview screen summarizes database locking activity for each attached process.

```

Node: GANDLF (1/3/24)   Oracle Rdb V7.3-13 Perf. Monitor 17-JUN-2014 14:44:32.23
Rate: 1.00 Second      Process Lock Overview           Elapsed: 160 00:32:42.98
Page: 1 of 1          USER_TEST:[DB_TESTING.TCS_MASTER]TCS.RDB;2      Mode: Global
-----
Process.ID Lock.Rqsts Prom.Rqsts Deadlocks. Blasted... Demotes... Unlocks...
2A46BE1F:1s      64      7519      0      214      7456      22
2A562A21:1s     1098     1355      0      247      346      1087
2A56002D:1A     8293     8803      7      2779     13331     5347
2A53DC2E:1A    10104    4359      8      2621     3581     5908
2A540E31:1A    18831    3409      7      788     13662    16509
2A532E37:1A     7337     2982      4      1694     2508     4109
2A52A43B:1A      0         0         0         0         0         0
2A536E3A:1A     6170     3836      0      1384     5604     3093
2A50C047:1A      0         0         0         0         0         0
2A4CDE49:1A      0         0         0         0         0         0
-----
Exit Help Menu >next_page <prev_page Options Set_rate Write Zoom !

```

The screen fields are the following:

- ◆ **Process.ID**
Identifies the attached database process. All processes on this screen contain the "A" suffix unless the process is a database server.
- ◆ **Lock.Rqsts**
This field gives the number of lock requests (also referred to as "enqueue" lock requests) for new locks. Whether the lock request succeeds or fails, it is included in this count.
- ◆ **Prom.Rqsts**
This field gives the number of enqueue lock requests to promote an existing lock to a higher lock mode. Whether or not the lock request succeeds, it is included in this count.
- ◆ **Deadlocks.**
This field gives the number of stalled enqueue lock requests for both new and promoted locks that ultimately resulted in a deadlock. Most deadlocks are tried again and resolved by Oracle Rdb without the application program ever knowing there was a deadlock. Therefore, the number shown in this field does not necessarily reflect the number of deadlocks reported to the application program.
- ◆ **Blasted...**
This field gives the number of blocking ASTs, sometimes referred to as "blasts", delivered to Oracle Rdb by the lock manager. A blocking AST is delivered to the holder of a lock when a lock conflict is detected, which is a good indication of contention problems. When Oracle Rdb receives a blocking AST, it often demotes or releases a lock in an attempt to avoid unnecessary deadlocks.
The number of blocking ASTs reported is composed of two different types of blocking ASTs: those generated externally and those generated internally.
An externally generated blocking AST occurs when a blocking AST is actually received by the process from the operating system in response to some lock conflict with another process. A blocking AST routine is executed and the RMU Show Statistics utility records the blocking AST activity.
An internally generated blocking AST occurs when a lock-blocking AST routine is executed by the process in anticipation that the same work would have to be performed anyway if a blocking AST were to be received from the operating system. This algorithm serves as an optimistic code optimization. The process, assuming it would eventually receive a blocking AST for the particular lock, executes the blocking AST routine. The RMU Show Statistics utility does not differentiate between these two types of blocking ASTs.
- ◆ **Demotes...**
This field gives the number of enqueue lock requests to demote an existing lock to a lower lock mode. These requests always succeed.
- ◆ **Unlocks...**
This field gives the number of deallocating lock requests to release an existing lock. These requests always succeed.

Process Object Overview Screen

The Process Object Overview screen summarizes rootfile object activity for each attached process. The rootfile objects are the KROOT, FILID, SEQBLK, TSNBLK, AIJDB, AIJFB, RTUPB, ACTIVE bitmap, CPT, RCACHE, CLIENT, UTILITY, and the CLIENT SEQUENCE.

```

Node: GANDLF (1/3/24)   Oracle Rdb V7.3-13 Perf. Monitor 17-JUN-2014 14:44:33.34
Rate: 1.00 Second           Process Object Overview           Elapsed: 160 00:32:44.09
Page: 1 of 1               USER_TEST:[DB_TESTING.TCS_MASTER]TCS.RDB;2       Mode: Global
-----
Process.ID Objs.Shrd. Objs.Excl. Objs.Rfrsh Objs.Updt. Objs.Write Objs.Relsd
2A46BE1F:1s   135255         25         52         24         24         135280
    
```

Oracle® Rdb for OpenVMS

2A562A21:1s	1497	48	103	20	20	1545
2A56002D:1A	6162	1712	107	510	510	7874
2A53DC2E:1A	5111	1620	70	479	479	6731
2A540E31:1A	1844	416	42	150	150	2260
2A532E37:1A	3190	953	61	297	297	4143
2A52A43B:1A	0	0	0	0	0	0
2A536E3A:1A	3959	1016	57	302	302	4975
2A50C047:1A	0	0	0	0	0	0
2A4CDE49:1A	0	0	0	0	0	0

 Exit Help Menu >next_page <prev_page Options Set_rate Write Zoom !

The screen fields are the following:

- ◆ **Process.ID**
Identifies the attached database process. All processes on this screen contain the "A" suffix unless the process is a database server.
- ◆ **Objs.Shrd**
This field displays the number of objects that are fetched for shared retrieval access.
- ◆ **Objs.Excl**
This field displays the number of objects that are fetched for exclusive access with the intention of subsequently being updated. This statistic does not indicate that the object was actually updated.
- ◆ **Objs.Rfrsh**
This field displays the number of objects whose information in the global section was detected as being stale, so the information was read again from the database root file.
- ◆ **Objs.Updt**
This field displays the number of objects whose information was modified. Only objects fetched for exclusive access can be modified.
- ◆ **Objs.Write**
This field displays the number of objects whose information was written back to the database root file.
- ◆ **Objs.Relsd**
This field displays the number of objects whose shared or exclusive access was released to other processes.

Process Transaction Overview Screen

The Process Transaction Overview screen summarizes transaction and checkpoint activity for each attached process.

Node: GANDLF (1/3/24) Oracle Rdb V7.3-13 Perf. Monitor 18-JUN-2014 13:09:48.19
 Rate: 1.00 Second Process Transaction Overview Elapsed: 160 22:57:58.94
 Page: 1 of 1 USER_TEST:[DB_TESTING.TCS_MASTER]TCS.RDB;3 Mode: Global

Process.ID	Transactns	TxCommitted	TxRollback	Checkpoint	VerbSucces	VerbFailur
2A565E77:1s	2	2	0	0	2	0
2A53B278:1s	1	1	0	0	1	0
2A55DC81:38	22	22	0	1	204	0
2A54348E:1A	4	4	0	0	73	0
2A562A89:1A	53	53	0	8	3056	0
2A520A84:1A	542	542	0	43	3357	5
2A43A88C:1A	489	489	0	40	3030	4
2A561A8D:1A	1	1	0	0	60	0
2A539A90:1A	1	1	0	0	60	0

Oracle® Rdb for OpenVMS

2A55FA8F:1A 1 1 0 0 60 0

Exit Help Menu >next_page <prev_page Options Set_rate Write Zoom !

The screen fields are the following:

- ◆ **Process.ID**
Identifies the attached database process. All processes on this screen contain the "A" suffix unless the process is a database server.
- ◆ **Transactns**
This field gives the number of completed database transactions. This is the count of the transaction COMMIT and ROLLBACK statements that have executed.
- ◆ **TxCommitted**
This field identifies the actual number of transactions that committed successfully to the database.
- ◆ **TxRollback**
This field identifies the actual number of transactions that aborted and were not applied to the database.
- ◆ **Checkpoint**
This field identifies the number of checkpoints performed by users. This field does not include the initial checkpoint when the user first attaches to the database.
- ◆ **VerbSucces**
This field gives the number of completed verbs that returned a successful status code. A verb is an atomic SQL statement or action. For example, a record insert is a verb, as is a record deletion.
Also, within a compound statement, each individual statement is atomic and Oracle Rdb performs a verb–success operation after processing each one. To avoid this overhead, you can use the SQL BEGIN ATOMIC compound statement to treat the entire block as a single verb.
- ◆ **VerbFailur**
This field gives the number of completed verbs that returned an error status code. Errors include end–of–collection and deadlocks, as well as all other exception conditions. A verb is an atomic SQL statement or action. For example, a record insert is a verb, as is a record deletion.
Excessive verb failures are usually an indication of a failed constraint, such as uniqueness criteria or an invalid data definition language (DDL) statement. Note that in the case of cursors and scans, reaching the end–of–stream always results in a verb failure. Note that SQL performs its own internal queries to identify metadata, such as relation or index names.
Oracle Rdb rarely issues a verb–failure unless there is an exception of some kind, such as a constraint failure.

Process Record Overview

The Process Record Overview screen summarizes database record activity for each attached process.

Node: GANDLF (1/3/24) Oracle Rdb V7.3-13 Perf. Monitor 17-JUN-2014 14:44:34.65
Rate: 1.00 Second Process Record Overview Elapsed: 160 00:32:45.40
Page: 1 of 1 USER_TEST:[DB_TESTING.TCS_MASTER]TCS.RDB;2 Mode: Global

Process.ID RecFetched Rec.Marked Rec.Stored Pag.Checkd PagDiscard Rec.Erased
2A46BE1F:1s 0 0 0 0 0
2A562A21:1s 0 0 0 0 0

Oracle® Rdb for OpenVMS

2A56002D:1A	276020	50497	23890	23890	0	0
2A53DC2E:1A	2676	475	448	448	0	0
2A540E31:1A	42975	12291	10341	10492	151	0
2A532E37:1A	1845	286	273	273	0	0
2A52A43B:1A	0	0	0	0	0	0
2A536E3A:1A	164922	32011	15046	15046	0	0
2A50C047:1A	0	0	0	0	0	0
2A4CDE49:1A	0	0	0	0	0	0

 Exit Help Menu >next_page <prev_page Options Set_rate Write Zoom !

The screen fields are the following:

- ◆ **Process.ID**
 Identifies the attached database process. All processes on this screen contain the "A" suffix unless the process is a database server.
- ◆ **RecFetched**
 This field gives the number of records fetched, including snapshot records. This field does not include records retrieved from a temporary table.
 Note that this value may be more than the actual number of records returned by a query. The reason is that queries may fetch records during the search phase and then re-fetch the selected records so that they may be returned to the user. Also, for uniform format storage areas, a sequential scan needs to fetch the next record on each page of the clump, even if there are no records on that page. In addition, every page in a uniform format storage area incurs an extra fetch to verify that there are no more records residing on that page.
- ◆ **Rec.Marked**
 This field gives the number of records marked. A record is marked when it is modified or erased, but not when it is stored. This field does not include records modified in a temporary table.
- ◆ **Rec.Stored**
 This field gives the number of records stored in the database. This field does not include records stored in temporary tables.
- ◆ **Pag.Checkd**
 This field indicates the number of pages checked in order to store a record. Ideally, very few candidate pages need to be checked when storing a record. However in certain cases, depending on record size, access method, locked space on a page, and SPAM thresholds, storing a record requires a number of page fetches.
- ◆ **Pag.Discard**
 This field identifies the number of pages checked but discarded because the actual free space on that page did not meet the physical requirements needed to store a new record.
- ◆ **Rec.Erased**
 This field gives the number of records erased from the database. This field does not include records erased from temporary tables.

Process Snapshot Overview Screen

The Process Snapshot Overview screen summarizes snapshot area activity for each attached process.

```

Node: GANDLF (1/3/24)   Oracle Rdb V7.3-13 Perf. Monitor 17-JUN-2014 14:44:36.44
Rate: 1.00 Second      Process Snapshot Overview      Elapsed: 160 00:32:47.19
Page: 1 of 1          USER_TEST:[DB_TESTING.TCS_MASTER]TCS.RDB;2      Mode: Global
-----
Process.ID SnpRecRtrv SnpLinFtch SnpPagRead SnpRecStor SnpPagFull SnpLckCnft
  
```


Oracle® Rdb for OpenVMS

2A46BE1F:1s	0	0	0	0	0	0
2A562A21:1s	0	0	0	0	0	0
2A56002D:1A	303	0	0	6581	916	0
2A53DC2E:1A	0	0	0	193	50	2
2A540E31:1A	303	0	0	3467	29	0
2A532E37:1A	0	0	0	157	38	0
2A52A43B:1A	0	0	0	0	0	0
2A536E3A:1A	303	0	0	70	1	0
2A50C047:1A	0	0	0	0	0	0
2A4CDE49:1A	0	0	0	0	0	0

Exit Help Menu >next_page <prev_page Options Set_rate Write Zoom !

The screen fields are the following:

- ◆ **Process.ID**
Identifies the attached database process. All processes on this screen contain the "A" suffix unless the process is a database server.
- ◆ **SnpRecRtrv**
This field gives the number of records retrieved by read-only transactions.
- ◆ **SnpLinFtch**
This field gives the number of lines fetched by read-only transactions. To retrieve a single record, a transaction might actually check a number of lines, some of which may be empty.
- ◆ **SnpPagRead**
This field gives the number of snapshot pages fetched by read-only transactions. If this count is high relative to the other read fields, read-only transactions are fetching records that are being updated frequently and the snapshot file is being used extensively.
- ◆ **SnpRecStor**
This field gives the number of records stored in the snapshot file by update transactions. Every snapshot record stored by an update transaction implies that a snapshot page was found and utilized. In the best case, this is a single-page fetch. The page in use, page too full, page conflict, and extended file sub-fields indicate some of the extra overhead involved in finding suitable snapshot pages on which to store snapshot records.
- ◆ **SnpPagFull**
This field gives the number of pages fetched that were unsuitable for storing snapshot records because there was not enough room on the snapshot page to include another version of a record. In this case, a new snapshot page must be fetched and linked with the full page. This allows read-only transactions to follow a chain of snapshot pages to find the correct version of a record.
- ◆ **SnpLckCnft**
This field gives the number of times a snapshot page fetch was requested but aborted due to a lock conflict with another process. When a page fetch conflicts with another process, another target page is fetched and checked so the lock conflict does not cost a disk I/O operation.

9.1.13 RMU Error Messages Which Suggest Altering Backup File Attributes

Enhancement Bug 10330130

Oracle Rdb database backup (RBF) files are created by the RMU/BACKUP command with specific OpenVMS RMS file attributes. If some of these attributes are then changed by ZIP or other file

Oracle® Rdb for OpenVMS

utilities which can change file attributes, the RMU commands RMU/RESTORE, RMU/RESTORE/ONLY_ROOT and RMU/DUMP/BACKUP which read Oracle Rdb database backup (RBF) files may not be able to read the backup file and will abort and return an error. When this happens, the OpenVMS SET FILE/ATTRIBUTE command can sometimes be used to alter the backup file attributes so that the RMU commands RMU/RESTORE, RMU/RESTORE/ONLY_ROOT and RMU/DUMP/BACKUP can then read the backup file.

Starting with Oracle Rdb V7.2–57 and Oracle Rdb V7.3–13, when the RMU commands RMU/RESTORE, RMU/RESTORE/ONLY_ROOT and RMU/DUMP/BACKUP cannot read a database backup file and the file attributes indicate that the problem may possibly be corrected by the OpenVMS SET FILE/ATTRIBUTE command, the RMU command will be aborted and one of the two following fatal error messages will be output to suggest that the OpenVMS SET FILE/ATTRIBUTE command may be able to make the backup file readable by the RMU command by altering the file attributes to a fixed record format and/or specifying the longest record length to be the suggested number of bytes. The longest record length is often 32256 bytes but may vary. Note that this is only a suggestion. Oracle Rdb cannot guarantee that this will make the backup file readable by the RMU command.

```
%RMU-F-INVBACKFIL, DISK:[DIRECTORY]FILENAME.EXT; is not a valid backup file
-RMU-I-INVFILATR, possibly use SET FILE/ATTRIBUTE=(RFM:FIX,LRL:32256) on this
backup file
```

or

```
%RMU-F-NOTBLKSIZ, invalid block size in backup file
-RMU-I-INVFILLRL, possibly use SET FILE/ATTRIBUTE=(LRL:32256) on this backup
file
```

The following example shows the fatal RMU-F-INVBACKFIL message put out when the RMU/DUMP/BACKUP command cannot read a backup file, followed by the informational RMU-I-INVFILATR message, which suggests the possibility that the OpenVMS SET FILE/ATTRIBUTE=(RFM:FIX,LRL:32256) command can be used to change the backup file attributes to the specified values so that the RMU/DUMP/BACKUP command can read the backup file. In this case, executing the suggested SET FILE/ATTRIBUTE=(RFM:FIX,LRL:32256) command does correct the problem and the RMU/DUMP/BACKUP command now succeeds. However, there is no guarantee that this will fix the problem.

```
$ RMU/DUMP/BACKUP DISK:[DIRECTORY]BACKUP_FILE.RBF
%RMU-I-DMPTXT_163, No dump option selected. Performing read check.
%RMU-F-INVBACKFIL, DISK:[DIRECTORY]BACKUP_FILE.RBF; is not a valid backup file
-RMU-I-INVFILATR, possibly use SET FILE/ATTRIBUTE=(RFM:FIX,LRL:32256) on this
backup file
%RMU-F-FATALERR, fatal error on DUMP_BACKUP
%RMU-F-FTL_DUMP, Fatal error for DUMP operation at 7-MAY-2015 17:01:15.03
$
$ SET FILE/ATTRIBUTE=(RFM:FIX,LRL:32256) DISK:[DIRECTORY]BACKUP_FILE.RBF
$
$ RMU/DUMP/BACKUP DISK:[DIRECTORY]BACKUP_FILE.RBF
%RMU-I-DMPTXT_163, No dump option selected. Performing read check.
```

The following example shows the fatal RMU-F-NOTBLKSIZ message put out when the RMU/DUMP/BACKUP command cannot read a backup file block, followed by the informational RMU-I-INVFILLRL message, which suggests the possibility that the OpenVMS SET FILE/ATTRIBUTE=(LRL:32256) command can be used to change the backup file attributes to the specified value so that the RMU/DUMP/BACKUP command can read the backup file. In this case,

Oracle® Rdb for OpenVMS

executing the suggested SET FILE/ATTRIBUTE=(LRL:32256) command does correct the problem and the RMU/DUMP/BACKUP command now succeeds. However, there is no guarantee that this will fix the problem. The same error is returned when the RMU/RESTORE command tries to read the same backup file and again in this case using the SET FILE/ATTRIBUTE command to change the backup file attributes allows RMU/RESTORE to read the backup file.

```
$ RMU/DUMP/BACKUP DISK:[DIRECTORY]BACKUP_FILE.RBF
%RMU-I-DMPTXT_163, No dump option selected. Performing read check.
%RMU-F-NOTBLKSIZ, invalid block size in backup file
-RMU-I-INVFILLRL, possibly use SET FILE/ATTRIBUTE=(LRL:32256) on this backup
file
%RMU-F-FATALERR, fatal error on DUMP_BACKUP
%RMU-F-FTL_DUMP, Fatal error for DUMP operation at 8-MAY-2015 13:25:25.61
$
$ SET FILE/ATTRIBUTE=(LRL:32256) DISK:[DIRECTORY]BACKUP_FILE.RBF
$
$ RMU/DUMP/BACKUP DISK:[DIRECTORY]BACKUP_FILE.RBF
%RMU-I-DMPTXT_163, No dump option selected. Performing read check.
$
$ RMU/RESTORE/DIRECTORY=DEVICE:[DIRECTORY]/NOCDD/NOLOG BACKUP_FILE.RBF
%RMU-F-NOTBLKSIZ, invalid block size in backup file
-RMU-I-INVFILLRL, possibly use SET FILE/ATTRIBUTE=(LRL:32256) on this backup
file
%RMU-F-FATALERR, fatal error on RESTORE
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 8-MAY-2015 13:26:15.44
$
$ SET FILE/ATTRIBUTE=(LRL:32256) DISK:[DIRECTORY]BACKUP_FILE.RBF
$
$ RMU/RESTORE/DIRECTORY=DEVICE:[DIRECTORY]/NOCDD/NOLOG BACKUP_FILE.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.
```

9.1.14 Query Optimization Improvements for DATE ANSI Queries

In prior releases of Oracle Rdb, a query such as the following would not use the index on the source column because the CAST function obscured the column reference from the optimizer.

The following example shows a query that is required to select all the transaction records that appeared on a specific date. That is, the query wants to ignore the time portion during the query.

```
SQL> select posting_timestamp
cont> from TRANSACTION_LOG
cont> where cast (posting_timestamp as DATE ANSI) = date ansi'2015-3-19'
cont> ;
Tables:
  0 = TRANSACTION_LOG
Index only retrieval of relation 0:TRANSACTION_LOG
Index name TRANS_NXD [1:1]
Keys: (0.POSTING_TIMESTAMP >= DATE '2015-03-19') AND (0.POSTING_TIMESTAMP <
      (DATE '2015-03-19' + INTERVAL '1' DAY))
POSTING_TIMESTAMP
19-MAR-2015 00:25:21.73
1 row selected
SQL>
```

The Oracle Rdb optimizer now detects that an index column is within the CAST function and rewrites such queries to expose the index column. As can be seen, this query now performs an index range retrieval for all values in the specified date/time range.

9.1.15 New RMU Dump Metadata_File Command

The new RMU Dump Metadata_File command dumps the metadata file created by the RMU Unload After_Journal command in a user readable format. The metadata file is created when using the /SAVE_METADATA qualifier of the RMU Unload After_Journal command.

The metadata file can also be dumped if both the /RESTORE_METADATA and the /OPTIONS=DUMP qualifiers are specified with the RMU Unload After_Journal command.

For complete information on the use and restrictions of the /SAVE_METADATA and /RESTORE_METADATA qualifiers, see the Oracle RMU Reference Manual section on the RMU Unload After_Journal command.

The RMU Dump Metadata_File command provides an independent command to dump the file and optionally outputs just the metadata file header record which identifies the database root file and version and other general information about the metadata file dump without having to dump the entire metadata file.

Note

Oracle Corporation reserves the right to change the format of the metadata file as required by future releases. The metadata file is intended to only be used internally by the RMU UNLOAD AFTER_JOURNAL command. New metadata files should be generated to reflect changing metadata definitions in the database. Metadata files incompatible with the current database version or metadata file version level will not be processed and will need to be recreated from the current database.

Syntax

The format of the RMU Dump Metadata_File command is:

```
RMU/DUMP/METADATA_FILE metadata-file-name
```

The default file type for the metadata-file-name is ".metadata".

Qualifiers

Additional qualifiers that can be used with this command are the following:

- ◆ /HEADER_ONLY
If this qualifier is specified, only the first header record in the metadata file will be output. The default if this qualifier is not specified is to output all records in the metadata file.
- ◆ /OUTPUT=output_filename
If this qualifier is specified, the metadata file will be dumped to the required named output file. The default file type for the output file is ".lis". The default if this qualifier is not specified is to output the metadata file records to the default output device.

Usage Notes

To use the RMU Dump Metadata_File command, you must have OpenVMS READ and READALL access to the metadata file to be dumped or be granted the OpenVMS SYSPRV or BYPASS privileges.

Examples

The following example first creates the TEST_DATABASE.METADATA file from the database using the /SAVE_METADATA qualifier of the RMU/UNLOAD AFTER_JOURNAL command. The RMU/DUMP/METADATA_FILE/HEADER_ONLY command is used to dump only the metadata file header information to the default output device. The RMU/DUMP/METADATA_FILE/OUTPUT command is then used to save the entire dump of the TEST_DATABASE.METADATA file to the TEST_DATABASE.LIS file.

```

$ RMU/UNLOAD/AFTER_JOURNAL DEVICE:[DIRECTORY]TEST_DATABASE.RDB -
  /SAVE_METADATA=DEVICE:[DIRECTORY]TEST_DATABASE.METADATA/NOLOG
$ RMU/DUMP/METADATA_FILE/HEADER_ONLY DEVICE:[DIRECTORY]TEST_DATABASE.METADATA
*-----*
* Oracle Rdb V7.3-2                               27-JUL-2015 16:02:56.29
*
* Dump of LogMiner Metadata File
*   Filename: DEVICE:[DIRECTORY]TEST_DATABASE.METADATA;1
*   Database: DEVICE:[DIRECTORY]TEST_DATABASE.RDB;1
*
*-----*

Created  7-JUL-2015 15:33:05.91 by JONES
Version 73.0 (Oracle Rdb V7.3-2) / Checksum 0043000D
Metadata file version level is 101
Database timestamp is 18-MAY-2015 12:01:05.75
Logical Area Information Count is 280

$ RMU/DUMP/METADATA_FILE/OUTPUT=DEVICE:[DIRECTORY]TEST_DATABASE.LIS -
  DEVICE:[DIRECTORY]TEST_DATABASE.METADATA
$

```

9.1.16 New REPLACE_ROWS Qualifier Added to RMU Load Command

This release of Oracle Rdb adds a /REPLACE_ROWS qualifier to the RMU Load command.

If a PRIMARY KEY exists for the table, then the REPLACE_ROWS qualifier will instruct Oracle Rdb to delete the row matching the primary key columns prior to replacing the whole row with new values from the source data file. If no PRIMARY KEY exists for the table, then the REPLACE_ROWS qualifier is ignored (i.e. no delete will be performed before inserting new data).

```

$ rmu/unload sql$database employees employees
%RMU-I-DATRECUNL, 100 data records unloaded 18-AUG-2015 22:32:10.48.
$ rmu/load/replace_rows sql$database employees employees
%RMU-I-DATRECREAD, 100 data records read from input file.
%RMU-I-DATRECSTO, 100 data records stored 18-AUG-2015 22:32:17.36.

```

Please refer to the SQL REPLACE Statement for a complete description of the affects of the /REPLACE_ROWS qualifier.

9.1.17 RMU/SET SHARED_MEMORY/SECTION_NAME

Bug 7238283

When Oracle Rdb databases are opened, global sections containing important and shared data get created and mapped into memory. A global section name, used to identify that memory, is dynamically generated based, in part, on the root file's device name. The same name persists from one database open to another, so long as the root remains on the same storage device. If the root is moved to a different device, the global section name will change.

The OpenVMS SYSMAN utility allows the DBA to reserve physical memory for use by those global sections that have been declared by Rdb to be "memory resident". Please refer to the OpenVMS System Management Utilities Reference Manual for a complete description of the RESERVED_MEMORY commands.

Reserving memory can be problematic for DBAs that want to utilize this feature but who also occasionally may have a need to move the root file to a different disk. To add or delete reserved memory, an AUTOGEN and system reboot is required.

Starting in this release, Oracle Rdb provides syntax that allows the DBA to set a global section name that will persist regardless of where the root file resides. This name is stored in the database root file until it is overwritten by another name or negated. The syntax is as follows:

```
$ RMU /SET SHARED_MEMORY /[NO]SECTION_NAME = secname dbname
```

where "secname" is a string that will be used as the common name portion for all memory resident global sections associated with this database. At runtime, Rdb will add a prefix to the common portion in order to distinguish the different global section types. Currently, there are three Rdb global section types that can be made memory resident:

- ◆ The database global section (TROOT/NODGBL),
- ◆ Row Cache global sections,
- ◆ Row Cache RUJ Buffers.

The prefix is composed using: "RDM" + (RDB major–minor version) + (a single character identifying the global section type). For example, in this current Oracle Rdb release, the prefixes would be:

- ◆ RDM73N for the database global section (TROOT/NODGBL),
- ◆ RDM73R for the Row Cache global sections,
- ◆ RDM73J for the Row Cache RUJ Buffers.

After using the above syntax to set the section name, you can see the actual complete name for each affected global section using the RMU/DUMP/HEADER/OPT=DEBUG command. For this example, all three global sections are set as memory–resident:

```
$ RMU/SET SHARED_MEMORY/SECTION_NAME=$$MFP MF_PERSONNEL
$ RMU/DUMP/HEADER/OPT=DEBUG/OUT=MFP.DMP MF_PERSONNEL
$ SEARCH MFP.DMP "GLOBAL SECTION NAME"
  - Global Section Name is "RDM73N$$MFP00000000"
```

Oracle® Rdb for OpenVMS

```
- Global Section Name is "RDM73J$$MFP00000000"
- Global Section Name is "RDM73R$$MFP00000001"
- Global Section Name is "RDM73R$$MFP00000002"
- Global Section Name is "RDM73R$$MFP00000003"
- Global Section Name is "RDM73R$$MFP00000004"
- Global Section Name is "RDM73R$$MFP00000005"
- Global Section Name is "RDM73R$$MFP00000006"
- Global Section Name is "RDM73R$$MFP00000007"
- Global Section Name is "RDM73R$$MFP00000008"
- Global Section Name is "RDM73R$$MFP00000009"
- Global Section Name is "RDM73R$$MFP0000000A"
- Global Section Name is "RDM73R$$MFP0000000B"
- Global Section Name is "RDM73R$$MFP0000000C"
- Global Section Name is "RDM73R$$MFP0000000D"
- Global Section Name is "RDM73R$$MFP0000000E"
- Global Section Name is "RDM73R$$MFP0000000F"

$
$ RMU/SET SHARED_MEMORY/NOSECTION_NAME MF_PERSONNEL
$ RMU/DUMP/HEADER/OPT=DEBUG/OUT=MFP.DMP MF_PERSONNEL
$ SEARCH MFP.DMP "GLOBAL SECTION NAME"
- Global Section Name is "RDM73N$1$DGA21235C714FC000000000000"
- Global Section Name is "RDM73J$1$DGA21235C714FC000000000000"
- Global Section Name is "RDM73R$1$DGA21235C714FC000000000001"
- Global Section Name is "RDM73R$1$DGA21235C714FC000000000002"
- Global Section Name is "RDM73R$1$DGA21235C714FC000000000003"
- Global Section Name is "RDM73R$1$DGA21235C714FC000000000004"
- Global Section Name is "RDM73R$1$DGA21235C714FC000000000005"
- Global Section Name is "RDM73R$1$DGA21235C714FC000000000006"
- Global Section Name is "RDM73R$1$DGA21235C714FC000000000007"
- Global Section Name is "RDM73R$1$DGA21235C714FC000000000008"
- Global Section Name is "RDM73R$1$DGA21235C714FC000000000009"
- Global Section Name is "RDM73R$1$DGA21235C714FC00000000000A"
- Global Section Name is "RDM73R$1$DGA21235C714FC00000000000B"
- Global Section Name is "RDM73R$1$DGA21235C714FC00000000000C"
- Global Section Name is "RDM73R$1$DGA21235C714FC00000000000D"
- Global Section Name is "RDM73R$1$DGA21235C714FC00000000000E"
- Global Section Name is "RDM73R$1$DGA21235C714FC00000000000F"
```

9.1.18 RESTART Clause of ALTER SEQUENCE No Longer Needs Value

With this release of Oracle Rdb, the ALTER SEQUENCE statement no longer requires a value for the RESTART clause. If the WITH literal value is omitted, then the existing START WITH (initial value) of the sequence will be used by the restart.

The following example shows the changed syntax and the result of the RESTART clause.

```
SQL> show sequence A;
A
Sequence Id: 1
Initial Value: 33
Minimum Value: 1
Maximum Value: 1000
Next Sequence Value: 93
Increment by: 1
Cache Size: 20
```

```

No Order
Cycle
No Randomize
Wait
SQL>
SQL> alter sequence A
cont>      restart;
SQL>
SQL> show sequence A;
      A
Sequence Id: 1
Initial Value: 33
Minimum Value: 1
Maximum Value: 1000
Next Sequence Value: 33
Increment by: 1
Cache Size: 20
No Order
Cycle
No Randomize
Wait
SQL>

```

9.1.19 New Options to SET SQLDA Statement

Starting in Oracle Rdb Release 7.3.1, the COUNT(*), COUNT (ALL value–expr) and COUNT (DISTINCT value–expr) functions returned BIGINT results. In some cases, applications using Dynamic SQL were not prepared to handle the larger type for COUNT. Therefore, in this release of Oracle Rdb, SQL allows the application to revert to accepting INTEGER counts.

```

enable-option =
+--> INSERT RETURNING +-->
|
|
+--> INTEGER COUNT -----+
|
|
+--> NAMED MARKERS -----+
|
|
+--> ROWID TYPE -----+

```

The following example uses Dynamic SQL and accepts various statements from the user. When using SET SQLDA, the returned data type of COUNT is altered from the default (BIGINT) to INTEGER.

```

-> ATTACH 'filename sql$database';
inputs: 0
-> SELECT COUNT(*) FROM RDB$DATABASE;
inputs: 0
out: [0] typ=Bigint {505} len=8
[SQLDA - displaying 1 fields]
0/: 1
-> SET SQLDA 'enable integer count';
inputs: 0
-> SELECT COUNT(*) FROM RDB$DATABASE;
inputs: 0
out: [0] typ=Integer {497} len=4
[SQLDA - displaying 1 fields]
0/: 1
-> EXIT;

```


Usage Notes

- ◆ **INTEGER COUNT** – The default behavior for Dynamic SQL is to expect the result data type of the COUNT function as BIGINT. When this option is enabled, Dynamic SQL will implicitly cast the result to INTEGER. If this option is disabled, then SQL will revert to a BIGINT result data type.

9.1.20 Support for External Authentication (LDAP)

Oracle Rdb now supports externally authenticated users when used with the USER clause of the ATTACH, ALTER DATABASE, CREATE DATABASE, CONNECT, DECLARE ALIAS, DROP DATABASE, and SET SESSION AUTHORIZATION statements.

```
SQL> attach 'filename APP_SYSTEM user -
cont>  'jenny.p.jones' using 'thepassword' ;
```

OpenVMS allows the system administrator to install and configure the ACME login and ACME authentication agents. Please refer to the HPE OpenVMS documentation for details and instructions. These services allow OpenVMS to use credentials from an external source (such as LDAP server) to authenticate the login.

Once enabled on the OpenVMS system, the system administrator can use the AUTHORIZE utility to mark selected users as being externally authorized (also see the description of the flag EXTAUTH). When these users log in to OpenVMS, they use their credentials from an LDAP server.

```
$ RUN SYS$SYSTEM:AUTHORIZE
UAF> MODIFY J_JONES /FLAGS=EXTAUTH
```

Oracle Rdb uses the Authentication and Credentials Management Extensions (ACME) subsystem through the SYS\$ACMW system service to also accept and authenticate user credentials through an LDAP server.

Note

Oracle Rdb returns the OpenVMS username when calling the built in functions CURRENT_USER, SESSION_USER and SYSTEM_USER. Therefore, when you login using an LDAP user, Rdb will use the mapping information (indirectly through SYS\$ACMW) that is defined in SYS\$STARTUP:LDAP_LOCALUSER_DATABASE.TXT to identify the user. This will include the username written to the after image journal.

Please see the HPE OpenVMS ACME LDAP Installation and Configuration Guide (SYS\$HELP:ACMELDAP_STD_CONFIG_INSTALL.TXT) for full details.

9.1.21 New RMU Extract Options to Control Output for DATABASE and ALTER_DATABASE Items

This release of Oracle Rdb adds the following new OPTIONS that control the output of the ALTER_DATABASE, DATABASE, and IMPORT items.

◆ JOURNALS

By default, /ITEM=ALTER_DATABASE will extract the after image journal settings and definitions for the database. Using /OPTION=NOJOURNALS will prevent that output.

◆ ROW_CACHES

By default, /ITEM=DATABASE or /ITEM=IMPORT will include the row cache definitions for the database. Using /OPTION=NOROW_CACHES will prevent that output.

Using /OPTION=ROW_CACHES with /ITEM=ALTER_DATABASE will include the cache definitions as ADD CACHE clauses.

◆ STORAGE_AREAS

By default, /ITEM=DATABASE or /ITEM=IMPORT will include the storage area definitions for the database. Using /OPTION=NOSTORAGE_AREAS will prevent that output.

Using /OPTION=STORAGE_AREAS with /ITEM=ALTER_DATABASE will include the storage area definitions as ADD STORAGE AREA clauses.

- ◆ In addition, the qualifier /OPTION=MATCH is now used to filter the storage area, row caches and journal names for ALTER_DATABASE, DATABASE, and IMPORT items.

Examples

The following example shows how to generate an ALTER DATABASE command that defines the three storage areas used for the EMPLOYEES table in the sample MF_PERSONNEL database. Here we disable the output of the journals, but request the storage areas with a matching wildcard string.

Example 9–2 Using the STORAGE_AREAS and JOURNALS options to control output

```
$ RMU/EXTRACT -
  /ITEM=ALTER_DATABASE -
  /DEFAULT=(NOALLOCATION,NOSNAPSHOT_ALLOCATION) -
  /OPTION=(NOHEADER,FILENAME_ONLY,ORDER_BY_NAME, -
           MATCH:EMPIDS%,STORAGE_AREAS,NOJOURNALS) -
  MF_PERSONNEL
```

This example shows the MATCH option being used to extract a row cache definition which is then used to recreate the row cache, possibly after database maintenance.

Example 9–3 Using MATCH option to extract just one row cache definition

```
$! use MATCH to just get the definition for the EMPLOYEES cache
$      RMU/EXTRACT -
      /ITEM=ALTER_DATABASE -
      RMU_EXTRACT_ALTER_DATABASE_EX -
      /OUTPUT=BEFORE.SQL -
      /DEFAULTS=(NOALLOCATION,NOSNAPSHOT_ALLOCATION) -
      /OPTION=(NOHEADER,FILENAME_ONLY,NOJOURNALS,ROW_CACHES,-
              MATCH=EMPLOYEES%)

$      SQL$
alter database
      filename RMU_EXTRACT_ALTER_DATABASE_EX

      drop cache EMPLOYEES
;
@BEFORE.SQL
SQL> set verify;
SQL> set language ENGLISH;
SQL> set default date format 'SQL92';
```

```

SQL> set quoting rules 'SQL92';
SQL> set date format DATE 001, TIME 001;
SQL> alter database
cont>     filename 'RMU_EXTRACT_ALTER_DATABASE_EX'
cont>
cont>     add cache EMPLOYEES
cont>         cache size is 300 rows
cont>         row length is 256 bytes
cont>         row replacement is DISABLED
cont>         shared memory is PROCESS
cont>         large memory is ENABLED
cont>         window count is 100
cont>         working set count is 10
cont>         number of reserved rows is 20
cont>         allocation is 100 blocks
cont>         extent is 100 blocks
cont>         row snapshot is ENABLED
cont>         (cache size is 900 rows)
cont> ; -- end alter database
$

```

9.1.22 RMU/SET AFTER_JOURNAL/SWITCH_JOURNAL Now Can Create an Emergency AIJ

Bug 21656497

The Oracle Rdb RMU/SET AFTER_JOURNAL/SWITCH_JOURNAL command changes the currently active After Image Journal (AIJ) file to the next available AIJ file if a fixed size AIJ journaling configuration is defined for an Rdb database. Normally, it is not necessary to use this command because the switch to the next available journal occurs automatically when the currently active fixed size AIJ file is full. However, the RMU/SET AFTER_JOURNAL/SWITCH_JOURNAL command can be used in cases where it is necessary to force a switch to the next available AIJ file, such as when it is necessary to switch to the next AIJ file on another disk when the disk used by the currently active fixed size AIJ file requires maintenance.

If a switch over to the next AIJ file cannot complete because the next AIJ file is not available (since it has not been backed up by the Automatic Backup Server (ABS) or for any other reason), the database enters the "AIJ suspended" state to avoid the loss of database data because it cannot be later recovered from an AIJ file. During this state, the database administrator can add new AIJ files or backup existing AIJ files to terminate the AIJ suspended state and allow suspended AIJ operations to continue.

Currently, if a database recovery (DBR) process is active during the AIJ suspended state, or a Hot Standby database replication process is active during the AIJ suspended state, or the AIJ Log Server (ALS) process is active and the RDM\$BIND_ALS_CREATE_AIJ system database bind logical is either not defined or defined as TRUE "1" and not FALSE "0", a new permanent "emergency" AIJ file will automatically be created for the switch over to terminate the AIJ suspended state. If for any reason an emergency journal cannot be created (because the maximum number of AIJ files defined for the database has already been reached or for any other reason), the AIJ suspended state will continue and the database administrator must resolve the situation or the database may be shut down (please see the Rdb AIJ related documentation for the complete details).

Functionality has been added to the RMU/SET AFTER_JOURNAL/SWITCH_JOURNAL command to allow it also to automatically create a permanent emergency AIJ journal file. The only way to prevent RMU/SET AFTER/SWITCH from creating an emergency journal is to explicitly define the system logical RDM\$BIND_ALS_CREATE_AIJ to be "0" in the LNM\$SYSTEM_TABLE.

As with emergency journals created in the already existing cases mentioned above, the emergency journals created by the RMU/SET AFTER_JOURNAL/SWITCH_JOURNAL command are permanent AIJ journals defined for the database. By default, they are created using the same device and directory as the currently active AIJ journal being switched from, unless the RDM\$BIND_AIJ_EMERGENCY_DIR database bind logical is defined to specify a different device and directory. Emergency AIJ journals are created using the same allocation definitions as the currently active AIJ journal being switched from. As currently, the generated name of the emergency AIJ is "EMERGENCY_XXX", where XXX is a series of 16 characters generated to create a unique name.

The following example shows this new feature. The RDM\$BIND_ALS_CREATE_AIJ logical has been defined as "1" in the LNM\$SYSTEM_TABLE to allow emergency AIJ journals to be created. This is also the default if the RDM\$BIND_ALS_CREATE_AIJ logical is not defined. If the RDM\$BIND_ALS_CREATE_AIJ logical is defined as "0", the RMU/SET AFTER_JOURNAL/SWITCH_JOURNAL command will not create an emergency AIJ file. The TEST database currently has two journals defined, "JOURNAL1" and "JOURNAL2", but additional AIJ slots are reserved in the database definition in case additional journals need to be created. The RMU/SET AFTER_JOURNAL/SWITCH_JOURNAL command is used to switch from "JOURNAL1" to "JOURNAL2". Then, when the RMU/SET AFTER_JOURNAL/SWITCH_JOURNAL command is used to switch from "JOURNAL2" back to "JOURNAL1", "JOURNAL1" is not available because it has not been backed up for some reason, perhaps because the Rdb AIJ AUTOMATIC BACKUP SERVER (ABS) is not running. The RMU/SET AFTER_JOURNAL/SWITCH_JOURNAL command automatically creates an emergency AIJ journal with the unique generated name "EMERGENCY_00B03639309BF694" and switches over to this permanent new database AIJ journal. Note that this is an exceptional case that only happens if no currently defined AIJ journal is available.

```
$ DEFINE/SYSTEM RDM$BIND_ALS_CREATE_AIJ 1
$ SHOW LOGICAL RDM$BIND_ALS_CREATE_AIJ
    "RDM$BIND_ALS_CREATE_AIJ" = "1" (LNM$SYSTEM_TABLE)
$
$ ! Put data in the first defined journal.
$
$ SQL$
ATTACH 'FILENAME TEST';
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
COMMIT;
EXIT;
$
$ ! Switch to the next defined journal.
$
```

Oracle® Rdb for OpenVMS

```
$ RMU/SET AFTER_JOURNAL/SWITCH_JOURNAL/LOG TEST
%RMU-I-OPERNOTIFY, system operator notification:
  After-image journal 0 switch-over in progress (to 1)
%RMU-I-OPERNOTIFY, system operator notification:
  Last unmodified AIJ journal has been selected
%RMU-I-OPERNOTIFY, system operator notification:
  After-image journal switch-over complete
%RMU-I-LOGMODSTR,      switching to after-image journal
  "JOURNAL2"
$
$ ! Put data in the next defined journal.
$
$ SQL$
ATTACH 'FILENAME TEST';
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
COMMIT;
EXIT;
$
$ ! Switch to the next journal.
$ ! An EMERGENCY journal with a generated
$ ! name such as "EMERGENCY_00B0333F5D37E224"
$ ! will be created since the existing
$ ! journals have not been backed up.
$
$ RMU/SET AFTER_JOURNAL/SWITCH_JOURNAL/LOG TEST
%RMU-I-OPERNOTIFY, system operator notification:
  After-image journal 1 switch-over in progress (to 2)
%RMU-I-OPERNOTIFY, system operator notification:
  Last unmodified AIJ journal has been selected
%RMU-I-OPERNOTIFY, system operator notification:
  After-image journal switch-over complete
%RMU-I-LOGMODSTR,      switching to after-image journal
  "EMERGENCY_00B03639309BF694"
$
$ SQL$
ATTACH 'FILENAME TEST';
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
COMMIT;
EXIT;
$
```

9.1.23 New Support for Second OpenVMS Account Password

Bug 627287

This release of Oracle Rdb supports connecting to a database with two passwords so that application environments which wish to use OpenVMS secondary password support can now participate with Rdb.

The following example shows the syntax that allows a second password to be specified.

```
SQL> connect to
cont>      'alias "ALIAS3" file date, file personnel'
cont>      as 'connection_name_3'
cont>      user 'prodsystem'
cont>      using (:password, :passwords)
cont> ;
SQL>
```

The following statements include this enhancement.

- ◆ ALTER DATABASE Statement
- ◆ ATTACH Statement
- ◆ CONNECT Statement (when part of the connect-expression or part of the ATTACH expression)
- ◆ CREATE DATABASE Statement
- ◆ DECLARE ALIAS Statement
- ◆ DROP DATABASE Statement
- ◆ EXPORT DATABASE Statement
- ◆ IMPORT DATABASE Statement
- ◆ SET SESSION AUTHORIZATION statement

Note

Remote access using two passwords requires the remote system to be running Oracle Rdb V7.3.2 or later.

9.1.24 New "Index Counts" Optimization for SORTED Indices

In prior releases of Oracle Rdb, a special optimization was applied to SORTED RANKED indices that reduced the I/O and CPU overhead for counting values within an index. In this release of Oracle Rdb, a similar optimization has been implemented for SORTED indices. The main benefit of this optimization is to greatly reduce the CPU overhead for processing SORTED indices with duplicate values.

The following example shows the new strategy applied for COUNT(*), COUNT(column), and COUNT(DISTINCT column). Here the column being referenced is the leading segment of a

SORTED index.

```
SQL> select count(*) from employees;
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (*) Q2
Index only retrieval of relation 0:EMPLOYEES
  Index name  MI_NDX [0:0]          Index counts

                        100
1 row selected
SQL> select count(middle_initial)
cont> from employees where middle_initial = 'A';
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (0.MIDDLE_INITIAL) Q2
Index only retrieval of relation 0:EMPLOYEES
  Index name  MI_NDX [1:1]          Index counts
  Keys: 0.MIDDLE_INITIAL = 'A'

                        4
1 row selected
SQL> select count(distinct middle_initial)
cont> from employees where middle_initial = 'A';
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (DISTINCT 0.MIDDLE_INITIAL) Q2
Index only retrieval of relation 0:EMPLOYEES
  Index name  MI_NDX [1:1]          Index distinct counts
  Keys: 0.MIDDLE_INITIAL = 'A'

                        1
1 row selected
SQL>
```

This optimization is enabled by default and controlled by the flag `COUNT_SCAN`. Use the `SET FLAGS 'NOCOUNT_SCAN'` statement to disable this optimization, if necessary.

9.1.25 Support for Proxy Access to Remote Databases Using TCP/IP Transport

Bugs 861258, 18106129 and 18246149

In prior releases of Oracle Rdb, proxy access to remote databases was supported only via OpenVMS and DECnet. This release of Oracle Rdb now also supports remote proxy access when using transport set to TCPIP.

Proxy access allows the system administrator to define the incoming node and user as being permitted to impersonate some other user. This new support in Rdb uses the same proxy database as used by OpenVMS DECnet proxy access. Therefore, if the environment is already established for DECnet proxy access, then very little is required to start using the support in Oracle Rdb when the transport is set to TCPIP.

The following changes will be required in such environments.

- ◆ Applications which attach using the DECnet node specification syntax that includes the target username.

For example, if the node specification looks like NODENAME"targetuser":: then this must be changed to use the USER clause of ATTACH, CONNECT, DECLARE ALIAS and similar statements that accept the database file specification. This is because the user and password specified as part of the node specification is part of the OpenVMS DECnet support and is not directly used by Oracle Rdb. So the following ATTACH statement needs to be changed to the updated format listed.

```
SQL> ATTACH 'filename lulu"targetuser"::dev:[dir]dbname';
```

```
SQL> ATTACH 'filename lulu::dev:[dir]dbname USER 'targetuser' ' ';
```

Alternately, the USER clause can be omitted and a configuration file can include the SQL_USERNAME parameter which will be used for the remote attach. If the proxy database on the target node includes this node and the specified user in the list of allowable proxies, then the connection will succeed.

Note

If DECnet proxy was previously used and the credentials (username) were not provided then, as with DECnet, the TCP/IP proxy lookup will expect a default entry for this node and the current user.

- ◆ A configuration file should be created at the system, group or user level that includes this line:

```
SQL_NETWORK_TRANSPORT_TYPE TCP/IP
```

This configuration parameter directs the Rdb/Dispatch layer of Rdb to use TCP/IP to connect to the remote database.

TCP/IP proxy access to Rdb databases is enabled by default if the remote node is also running Oracle Rdb V7.3.2.0 or later. If the system administrator does not wish to allow proxy access via TCP/IP, then the following parameter can be defined.

```
SQL_ENABLE_TCPIP_PROXY FALSE
```

This can appear on the client in the RDB\$CLIENT_DEFAULTS.DAT file or on the server in the RDB\$SERVER_DEFAULTS.DAT.

Please refer to *Oracle Rdb for OpenVMS Installation Guide* for more information about remote access and configuration files.

9.1.26 Support for INTEGER Result Type for COUNT Function

Bug 22313534

In Oracle Rdb V7.3.1, the COUNT(*), COUNT(ALL value_expression), and COUNT(DISTINCT value_expression) functions return BIGINT results. In some applications, this size integer is not handled or has special meaning to the application and therefore, with this release of Rdb, a new option is provided for the database so that SQL will revert to use INTEGER count results.

The following example show this.

```
SQL> alter database
cont> filename MF_PERSONNEL
cont> set count result as integer
cont> ;
```

The clause SET COUNT RESULT [AS] { INTEGER | BIGINT } can be specified on the CREATE DATABASE, ALTER DATABASE and IMPORT DATABASE statement. When the setting is INTEGER, that will be indicated by the SHOW DATABASE statement in Interactive SQL.

This flag will result in two changes:

1. Any database definition language (DDL) commands that derive data type results from expressions will now assume COUNT returns INTEGER. This includes: AUTOMATIC AS and COMPUTED BY columns, and columns of a CREATE VIEW or ALTER VIEW statement.
2. SQL applications will now also derive data types in a similar way to prior releases of Oracle Rdb. Namely, Interactive SQL will return INTEGER results from COUNT functions and Dynamic SQL will return an integer type (SQLDA_INTEGER) in the SQLDA or SQLDA2 when executing a DESCRIBE statement.

Applications written in SQL Module Language or using the SQL Pre-compiler will need to be recompiled after the database has been altered. In particular, if the application uses the COMPILETIME clause on the DECLARE ALIAS statement, the referenced database must have been altered with the SET COUNT RESULT AS INTEGER clause. If Dynamic SQL is being used, the type information will be derived from the target (RUNTIME) database so it must also be altered with SET COUNT RESULT AS INTEGER clause.

Use the ALTER VIEW statement to correct any views which were created with Oracle Rdb V7.3.1 and later. The EXPORT DATABASE and IMPORT DATABASE statement will preserve this setting.

Chapter 10

Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.3

10.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.3

10.1.1 Oracle Rdb 7.3.1.3 Certified on OpenVMS 8.4–1H1 from VMS Software Inc. and Integrity i4 systems from HPE

This version of Rdb, Release 7.3.1.3, has been certified to run on OpenVMS Version 8.4–1H1 from VMS Software Inc. on Integrity i4 systems from HPE.

Chapter 11

Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.2

11.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.2

11.1.1 New FULBCKREQ Message Output When a Full Backup is Required

Bug 18328148

There are some Oracle Rdb database changes that require the next database backup to be a full backup to guarantee correct database recovery using an incremental backup. If an incremental database backup is executed without a preceding full database backup, the incremental backup will be aborted with a fatal error.

```
$ rmu/backup/incremental/nolog mf_personnel.rdb -  
  DEVICE:[DIRECTORY]mfp.rbf  
%RMU-F-NOFULLBCK, no full backup of this database exists  
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 14-MAR-2014 13:45:21.35
```

A dump of the database header will show if this root flag is set.

```
$ rmu/dump/header mf_personnel  
*-----  
* Oracle Rdb V7.3-12                               14-MAR-2014 13:45:18.51  
*  
* Dump of Database header  
*   Database: DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1  
*  
*-----  
Database Parameters:  
  
  Database Backup...  
  
    - Incremental backup not allowed until full backup
```

A new warning message will now be output at the end of an ALTER DATABASE command if the ALTER DATABASE command contains one or more operations which require the next database backup to be a full database backup.

```
%RDMS-W-FULBCKREQ, The next database backup must be a full backup
```

These are the operations that require the next database backup to be a full database backup.

- ◆ Add one or more storage areas to the database.

```
$ SQL$  
alter data filename mf_personnel  
  add storage area new_area;  
%RDMS-W-FULBCKREQ, The next database backup must be a full backup  
$
```

- ◆ Delete one or more storage areas from the database.

```
$ SQL$
alter database filename mf_personnel
  drop storage area area1
  drop storage area area2
  drop storage area area3
  drop storage area area4;
%RDMS-W-FULBCKREQ, The next database backup must be a full backup
$
```

- ◆ Reserve database entries for additional storage areas.

```
$ SQL$
alter database filename mf_personnel
  reserve 10 storage areas;
%RDMS-W-FULBCKREQ, The next database backup must be a full backup
$
```

- ◆ Modify the live storage area page allocation.

```
$ SQL$
alter database filename mf_personnel
  alter storage area jobs allocation is 2000 pages;
%RDMS-W-FULBCKREQ, The next database backup must be a full backup
$
```

- ◆ Modify the maximum number of database users.

```
$ SQL$
alter database filename mf_personnel
  number of users is 50;
%RDMS-W-FULBCKREQ, The next database backup must be a full backup
$
```

- ◆ Modify the maximum number of database cluster nodes.

```
$ SQL$
alter database filename mf_personnel
  number of cluster nodes is 4;
%RDMS-W-FULBCKREQ, The next database backup must be a full backup
```

11.1.2 New TRACE Option for EXPORT DATABASE Statement

This release of Oracle Rdb adds a TRACE option to the EXPORT DATABASE Statement. This option enables tracing of certain operations internal to EXPORT. For example, when COMPRESSION and TRACE are specified, the TRACE option causes output of the compression percentages for each table, null bit vector (NBV) and list of byte varying data.

```
SQL> export database filename personnel into pers compression trace;
** compress nbv : <CANDIDATES> too small to compress
** compress data: <CANDIDATES> input 846 output 244 deflate 72%
** compress nbv : <COLLEGES> too small to compress
** compress data: <COLLEGES> input 896 output 556 deflate 38%
** compress nbv : <DEGREES> too small to compress
** compress data: <DEGREES> input 4785 output 2268 deflate 53%
** compress nbv : <DEPARTMENTS> too small to compress
** compress data: <DEPARTMENTS> input 1222 output 750 deflate 39%
** compress data: <EMPLOYEES> input 11700 output 4559 deflate 62%
** compress nbv : <EMPLOYEES> input 1200 output 808 deflate 33%
** compress nbv : <JOBS> too small to compress
** compress data: <JOBS> input 495 output 434 deflate 13%
```

Oracle® Rdb for OpenVMS

```
** compress nbv : <JOB_HISTORY> too small to compress
** compress data: <JOB_HISTORY> input 9316 output 6095 deflate 35%
** compress nbv : <RESUMES> too small to compress
** compress data: <RESUMES> too small to compress
** compress nbv : <SALARY_HISTORY> too small to compress
** compress data: <SALARY_HISTORY> input 18225 output 13695 deflate 25%
** compress nbv : <WORK_STATUS> too small to compress
** compress data: <WORK_STATUS> input 69 output 66 deflate 5%
SQL>
```

In this example, several tables and null bit vectors (NBV) can not be reduced by compression because of their small size.

11.1.3 New /NOAFTER_JOURNAL Qualifier to Disable AIJ File Creation by RMU/RECOVER

Bug 6656199

Oracle Rdb normally writes information to the after image journal (AIJ) file describing the creation of new after image journal files. There are cases where the user does not want RMU/RECOVER to process this information and recreate AIJ files, such as lack of disk space. This release of Oracle Rdb adds a new /NOAFTER_JOURNAL qualifier to the RMU/RECOVER command. If this qualifier is specified, no new Rdb AIJ files will be created by the current RMU/RECOVER command and any AIJ file deletion records for those AIJ files which were not created will also be ignored.

The syntax for this new RMU/RECOVER qualifier is as follows.

```
/[NO]AFTER_JOURNAL
```

The default if this qualifier is not specified is /AFTER_JOURNAL. Therefore, /NOAFTER_JOURNAL must be specified to ignore the creation of new AIJ files recorded in any AIJ file being recovered by the current RMU/RECOVER command.

In the following example, the creation of the new after image journal file J2.AIJ for the MF_PERSONNEL database is journaled to the current after image journal file RMU_RECOVER_4.AIJ_1. When the MF_PERSONNEL database is deleted and then restored from the MF_PERSONNEL.RBF file and then recovered from RMU_RECOVER_4.AIJ_1 using the new /NOAFTER_JOURNAL qualifier, the J2.AIJ file does not get created.

```
$!
$! Change the database to enable after image journaling to the
$! RMU_RECOVER_4.AIJ_1 AIJ file
$!
$ SQL$
    alter database filename MF_PERSONNEL
    reserve 10 journals
    journal filename disk:[directory]RMU_RECOVER_4.AIJ_1;
%RDMS-W-DOFULLBCK, full database backup should be done to ensure future recovery
    exit
$!
$! Backup the database
$!
$ RMU/BACKUP/NOLOG MF_PERSONNEL DISK:[DIRECTORY]MF_PERSONNEL.RBF
$!
```

Oracle® Rdb for OpenVMS

```
$! Add a new AIJ file J2.AIJ to put a create AIJ file record in the current
$! RMU_RECOVER_4.AIJ_1 AIJ file
$!
$ SQL$
  alter database file mf_personnel
      add journal j2 filename disk:[directory]j2 allocation is 1000 blocks;
exit
$!
$! Drop the database and then restore the database from the backup RBF file
$!
$ SQL$
  drop database filename MF_PERSONNEL;
  exit
$!
$! Delete the added j2.aij file
$!
$ DELETE DISK:[DIRECTORY]J2.AIJ;*
$!
$! Restore the database
$!
$ RMU/RESTORE/NOADD/NOLOG/NOAFTER_JOURNAL -
  /ROOT=DISK:[DIRECTORY]MF_PERSONNEL.RDB DISK:[DIRECTORY]MF_PERSONNEL.RBF
$!
$! Recover the database from RMU_RECOVER_4.AIJ_1 to show that the J2.AIJ
$! file does not get created if RMU/RECOVER/NOAFTER_JOURNAL is specified
$!
$ RMU /RECOVER /NOAFTER_JOURNAL /NOLOG /ROOT=DISK:[DIRECTORY]MF_PERSONNEL.RDB -
  DISK:[DIRECTORY]RMU_RECOVER_4.AIJ_1
%RMU-I-LOGRECDB, recovering database file DISK:[DIRECTORY]MF_PERSONNEL.RDB;1
%RMU-I-AIJSUCCESS, 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
$ DIR DISK:[DIRECTORY]J2.AIJ
%DIRECT-W-NOFILES, no files found
```

11.1.4 Enhance Dumper of Merge Range List

Bug 18530761

In prior releases of Oracle Rdb, the strategy display for a query with OR predicates could be misleading when multiple range lists were merged. The following example demonstrates this problem with a query performed with IN and OR predicates to restrict values to selected ranges.

```
create table TEST_TABLE
  (a char);
create index TEST_TABLE_INDEX on TEST_TABLE (A);

select * from TEST_TABLE
where a in ('A', 'A', 'V', 'B') or a between 'C' and 'Y';
Tables:
  0 = TEST_TABLE
Conjunct: (0.A = 'A') OR (0.A = 'V') OR (0.A = 'B') OR ((0.A >= 'C') AND (0.A
  <= 'Y'))
Index only retrieval of relation 0:TEST_TABLE
  Index name  TEST_TABLE_INDEX [(1:1)4]
  Keys: r0: (0.A >= 'C') AND (0.A <= 'Y')
  r1: 0.A = 'B'
```



```

      r2: 0.A = 'V'
      r3: 0.A = 'A'
0 rows selected

```

Rdb has merged the OR ranges into a single range list ('A' .. 'Y') and eliminated duplicate ranges. However, the STRATEGY and DETAIL display do not reflect this state.

In this release, use the SET FLAGS 'MERGE_RANGE_LIST' flag in addition to STRATEGY and DETAIL to display further details.

```

! turn on the display of merge_range_list
set flags 'merge_range_list';

select * from TEST_TABLE
where a in ('A', 'A', 'V', 'B') or a between 'C' and 'Y';
Tables:
  0 = TEST_TABLE
Conjunct: (0.A = 'A') OR (0.A = 'V') OR (0.A = 'B') OR ((0.A >= 'C') AND (0.A
          <= 'Y'))
Index only retrieval of relation 0:TEST_TABLE
Index name  TEST_TABLE_INDEX [(1:1)4]
  Keys: r0: (0.A >= 'C') AND (0.A <= 'Y')
        r1: 0.A = 'B'
        r2: 0.A = 'V'
        r3: 0.A = 'A'
Index name  TEST_TABLE_INDEX [1:1]
Columns: r0:{(0.A),(0.A)}
IKeys:   r0:{('A'),('Z')}
0 rows selected

```

Note that the upper range is encoded as a higher value internally so that the index scan is terminated correctly.

11.1.5 RMU Extract Now Extracts SYS_GET_DIAGNOSTIC Function

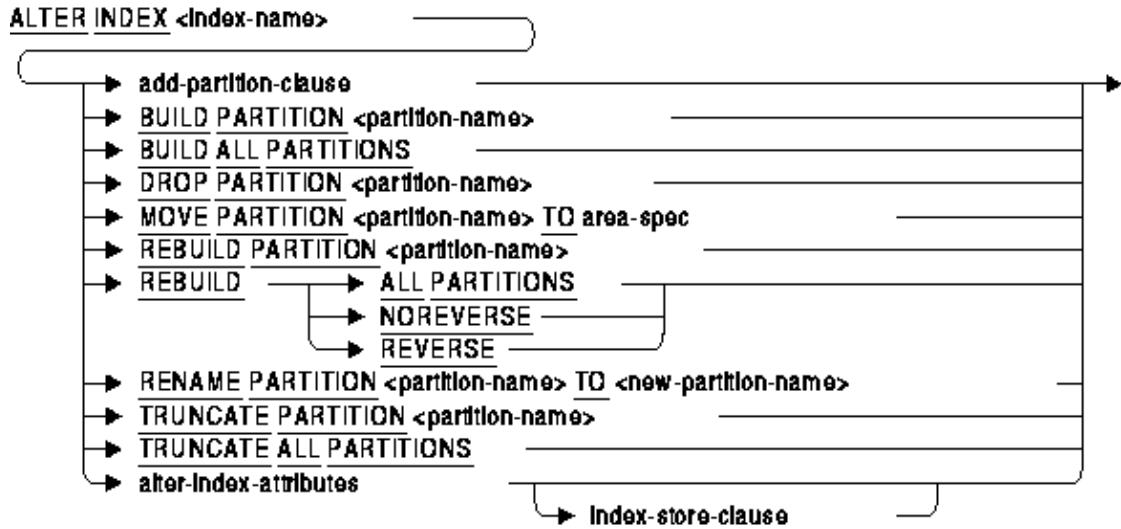
With this release of Oracle Rdb, the RMU Extract command now correctly formats the SYS_GET_DIAGNOSTIC function, which was added in earlier releases.

11.1.6 Alter Index Now Supports REVERSE and NOREVERSE Clauses

This release of Oracle Rdb now supports the REVERSE keyword on the ALTER INDEX statement as part of the REBUILD action. This clause requests that the named index be rebuilt as a REVERSE key index.

Syntax

The revised syntax for the ALTER INDEX statement is:



REVERSE
NOREVERSE

An existing SORTED or SORTED RANKED index can be converted to a REVERSE index by using this variation of the REBUILD ALL PARTITIONS clause. An already REVERSE index can be changed to a non-REVERSE index using the NOREVERSE keyword.

The following example shows an existing index being converted to a REVERSE index.

```

SQL> alter index DOCUMENTS
cont> rebuild reverse;
SQL>
  
```

Usage Notes

- ◆ If an index is already defined as REVERSE, then REBUILD REVERSE is equivalent to REBUILD ALL PARTITIONS.
- ◆ If an index is not defined as REVERSE, then REBUILD NOREVERSE is equivalent to REBUILD ALL PARTITIONS.
- ◆ The clause REVERSE may not be applied to a HASHED index.
- ◆ When applying REVERSE to an existing index, any column defined as DESC will be modified to remove the descending ordering.

11.1.7 SQL Precompiler Now Generates C++ Compatible Intermediate C Source

Enhancement Bug 1504425

This release of Oracle Rdb includes some support for using SQL Precompiler with the C++ compiler (CXX). The Rdb SQL precompiler now generates C++ compatible definitions when processing embedded SQL commands in a .SC source. This support does not support the C++ language, and the processed .SC file must conform to a legal C source format and language features.

Please note the following changes:

- ◆ The SQL\$PRE command line now accepts CXX as an option to the /CC qualifier. This option will direct the SQL precompiler to generate C code which is acceptable to the C++ compiler.
- ◆ The CXX DCL command will be used to invoke the C++ compiler, instead of the CC command. Additional qualifiers on the SQL\$PRE command line will be passed to the CXX compiler and must be legal qualifiers for C++.
- ◆ Function prototypes will include parameter definitions
- ◆ Function prototypes are enclosed by extern "C" to prevent the names being interpreted as C++ routines.

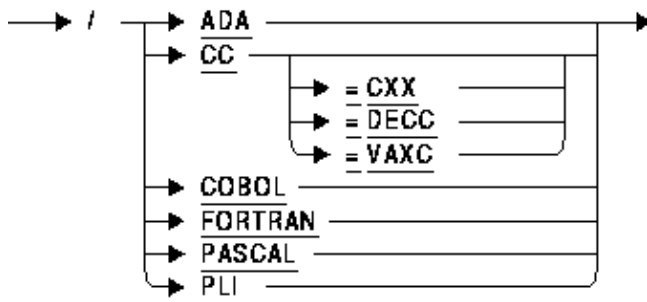
```
#ifdef __cplusplus
extern "C" {
#endif /* __cplusplus */
...
#ifdef __cplusplus
}
#endif /* __cplusplus */
```

- ◆ SQLCODE is expected to be defined as type *int*.
- ◆ On OpenVMS Alpha systems, the application is expected to be linked with the special library SYS\$LIBRARY:LIBCXXSTD.OLB. Please refer to the relevant HPE C++ documentation.

Syntax

The revised syntax for the PRE-LANG-QUALIFIERS qualifier is:

pre-lang-qualifiers =



11.1.8 New RMU/RECOVER Feature to Alter Storage Areas, After_Journal and Record_Cache Directories

Enhancement Bug 867636

Oracle® Rdb for OpenVMS

A new feature has been added to RMU/RECOVER and RMU/DUMP to alter the directory specifications for new storage areas, after image journal files and row caches created during the recovery of an Oracle Rdb database. This is only for the creation of new storage areas, after image journal files and row caches as recorded in the after image journal files used for a database recovery by the RMU/RECOVER command, and only for modifying the directory specifications defined in the after image journal files for the new storage areas, after image journal files and row caches at the time RMU/RECOVER creates them.

This new feature will allow the user to control where newly created storage areas, after image journal files and row caches are located. Previously, they could only be put in the directory locations recorded in the after image journal files used for the recovery.

Logicals can be used for the directory specifications but must translate to valid directory specifications that exist when the RMU/RECOVER or RMU/DUMP command is executed.

A new RMU/RECOVER /DIRECTORY qualifier can be used to specify one directory specification for all storage areas created, one directory specification for all after image journal files created and/or one directory specification for all row caches created.

The syntax for this new qualifier for storage areas is:

```
/DIRECTORY=AREAS=directory_specification
```

The syntax for this new qualifier for after image journal files is:

```
/DIRECTORY=AFTER_JOURNAL=directory_specification
```

The syntax for this new qualifier for row caches is:

```
/DIRECTORY=ROW_CACHE=directory_specification
```

To specify two or more of these options with the /DIRECTORY qualifier, use the following syntax.

```
/DIRECTORY=(AREAS=directory_specification, -  
AFTER_JOURNAL=directory_specification, -  
ROW_CACHE=directory_specification)
```

The following example shows all three options used with the /DIRECTORY qualifier in the RMU/RECOVER command.

```
$ RMU/RECOVER/NOLOG/NOTRACE/NOCONFIRM/NOAUTOMATIC-  
/DIRECTORY=(AREAS=DISK:[DIRECTORY], -  
AFTER_JOURNAL=DISK:[DIRECTORY], -  
ROW_CACHE=DISK:[DIRECTORY]) -  
DISK:[DIRECTORY]:TEST_J1_BCK.AIJ
```

To specify different directory specifications for specific storage areas, after journal files and row caches, option files can be designated using the new RMU/RECOVER /OPTIONS qualifier. One option file must be specified to select one or more storage areas for directory modification, one option file must be specified to select one or more after image journal files for directory modification and one option file must be specified to select one or more row caches for directory modification.

The syntax for this new qualifier for storage areas is:

Oracle® Rdb for OpenVMS

```
/OPTIONS=AREAS=option_file_specification
```

The syntax for this new qualifier for after image journal files is:

```
/OPTIONS=AFTER_JOURNAL=option_file_specification
```

The syntax for this new qualifier for row caches is:

```
/OPTIONS=ROW_CACHE=option_file_specification
```

To specify two or more of these options with the /OPTIONS qualifier, use the following syntax.

```
/OPTIONS=(AREAS=option_file_specification, -  
          AFTER_JOURNAL=option_file_specification, -  
          ROW_CACHE=option_file_specification)
```

The following example shows all three options used with the /OPTIONS qualifier in the RMU/RECOVER command.

```
$ RMU/RECOVER/NOLOG/NOTRACE/NOCONFIRM/NOAUTOMATIC-  
  /OPTIONS=(AREAS=DISK:[DIRECTORY]RECOVPTAREAS.OPT, -  
  AFTER_JOURNAL=DISK:[DIRECTORY]RECOVPTAIJ.OPT, -  
  ROW_CACHE=DISK:[DIRECTORY]RECOVPTRCACHE.OPT) -  
  DISK:[DIRECTORY]TEST_J1_BCK.AIJ
```

A new /DIRECTORY_OPTIONS qualifier has been added to the RMU/DUMP command to create storage area, after image journal and row cache option files for use with the /OPTIONS qualifier in the RMU/RECOVER command. These option files will contain entries for all storage areas, after image journal files and row caches defined for the database. These option files can then be edited by the user to eliminate storage area, after image journal or row cache entries that will not be created during the RMU/RECOVER session, or they can be used without being edited since entries for storage areas, after image journal files and row caches not created during the recovery operation will be ignored.

The syntax for this new qualifier for storage areas is:

```
/DIRECTORY_OPTIONS=AREAS=option_file_specification
```

The syntax for this new qualifier for after image journal files is:

```
/DIRECTORY_OPTIONS=AFTER_JOURNAL=option_file_specification
```

The syntax for this new qualifier for row caches is:

```
/DIRECTORY_OPTIONS=ROW_CACHE=option_file_specification
```

To specify two or more of these options with the /DIRECTORY_OPTIONS qualifier, use the following syntax.

```
/DIRECTORY_OPTIONS=(AREAS=option_file_specification, -  
                   AFTER_JOURNAL=option_file_specification, -  
                   ROW_CACHE=option_file_specification)
```

Oracle® Rdb for OpenVMS

The following example shows all three options used with the /DIRECTORY_OPTIONS qualifier in the RMU/DUMP command.

```
$ RMU/DUMP-  
  /NOHEADER-  
  /DIRECTORY_OPTIONS=(AREAS=DISK:[DIRECTORY]RECOVPTAREAS.OPT, -  
  AFTER_JOURNAL=DISK:[DIRECTORY]RECOVPTAJ.OPT, -  
  ROW_CACHE=DISK:[DIRECTORY]RECOVOPTRCACHE.OPT) -  
  DISK:[DIRECTORY]TEST
```

The following example, created by the new RMU/DUMP /DIRECTORY_OPTIONS qualifier, shows the format of the option file used with the new RMU/RECOVER /OPTIONS qualifier for storage areas. The live data storage area name is followed by /DIRECTORY=DISK:[DIRECTORY] for the live storage area file to specify the directory specification for the storage area *.RDA file. This is followed by /SNAPSHOT_DIRECTORY=DISK:[DIRECTORY] for the storage area snapshot file to specify the directory specification for the storage area *.SNP file. If /SNAPSHOT_DIRECTORY is not specified, the directory specification specified by /DIRECTORY is used for both the live and snapshot storage area files.

```
! Recover Areas Directory Options file for database  
! DISK:[DIRECTORY]FILENAME.EXT;VERSION  
! Created 22-JUL-2014 09:37:24.29  
! Created by DUMP command
```

```
RDB$SYSTEM -  
  /directory=DISK:[DIRECTORY] -  
  /snapshot_directory=DISK:[DIRECTORY]
```

```
TEST_A1 -  
  /directory=DISK:[DIRECTORY] -  
  /snapshot_directory=DISK:[DIRECTORY]
```

```
TEST_A2 -  
  /directory=DISK:[DIRECTORY] -  
  /snapshot_directory=DISK:[DIRECTORY]
```

```
TEST_A3 -  
  /directory=DISK:[DIRECTORY] -  
  /snapshot_directory=DISK:[DIRECTORY]
```

```
TEST_A4 -  
  /directory=DISK:[DIRECTORY] -  
  /snapshot_directory=DISK:[DIRECTORY]
```

The following example, created by the new /DIRECTORY_OPTIONS qualifier of the RMU/DUMP command, shows the format of the option file used with the new RMU/RECOVER /OPTIONS qualifier for after image journal files. The after image journal file name is followed by /DIRECTORY=DISK:[DIRECTORY] to specify the after image journal file directory specification. This is followed by /BACKUP_DIRECTORY=DISK:[DIRECTORY] to specify the directory specification for the after image journal backup file. If /BACKUP_DIRECTORY is not specified, the directory specification specified by /DIRECTORY is used for both the after image journal file and the after image journal backup file. If no backup directory is defined for the after image journal entry in the database, the backup directory specification will be ignored.

Oracle® Rdb for OpenVMS

```
! Recover After Journal Options file for database
! DISK:[DIRECTORY]FILENAME.EXT;VERSION
! Created 22-JUL-2014 09:37:24.29
! Created by DUMP command
```

```
TEST_J1 -
  /directory=DISK:[DIRECTORY] -
  /backup_directory=DISK:[DIRECTORY]
```

```
TEST_J2 -
  /directory=DISK:[DIRECTORY] -
  /backup_directory=DISK:[DIRECTORY]
```

```
TEST_J3 -
  /directory=DISK:[DIRECTORY] -
  /backup_directory=DISK:[DIRECTORY]
```

```
TEST_J4 -
  /directory=DISK:[DIRECTORY] -
  /backup_directory=DISK:[DIRECTORY]
```

The following example, created by the new `/DIRECTORY_OPTIONS` qualifier of the `RMU/DUMP` command, shows the format of the option file used with the new `RMU/RECOVER /OPTIONS` qualifier for row caches. The row cache name is followed by `/DIRECTORY=DISK:[DIRECTORY]` to specify the row cache directory specification. If no directory specification is defined for the row cache entry in the database, the directory specification will be ignored.

```
! Recover Row Cache Directory Options file for database
! DISK:[DIRECTORY]FILENAME.EXT;VERSION
! Created 22-JUL-2014 09:37:24.29
! Created by DUMP command
```

```
TEST_A1 -
  /directory=DISK:[DIRECTORY]
```

```
TEST_A2 -
  /directory=DISK:[DIRECTORY]
```

```
TEST_A3 -
  /directory=DISK:[DIRECTORY]
```

```
TEST_A4 -
  /directory=DISK:[DIRECTORY]
```

In the following example for database `DISK:[HOME]TEST.RDB`, the first `RMU/RECOVER` command uses the `/DIRECTORY` qualifier to change all directories from `DISK:[HOME]` to `DISK:[NEW]` for the storage areas `TEST_A3` and `TEST_A4`, the after image journal files `TEST_J3` and `TEST_J4` and the row caches `TEST_A3` and `TEST_A4`, whose creation was recorded in the journal file `TEST_J1_BCK.AIJ`. Then, the new `/DIRECTORY_OPTIONS` qualifier in the `RMU/DUMP` command is used to create option files for all `TEST` database after image journal files, storage areas and row caches, including those with directories changed to `DISK:[NEW]`. The `TEST`

Oracle® Rdb for OpenVMS

database is then deleted and again restored with all directories again set to DISK:[HOME]. The second RMU/RECOVER command then uses the /OPTIONS qualifier to specify the option files created by RMU/DUMP, which include the after image journal file, storage area and row cache directories changed by the first RMU/RECOVER to DISK:[NEW], to change the directories back again from the journaled directory specification of DISK:[HOME] to DISK:[NEW].

```
$ ! Create the TEST database
$
$ SQL
CREATE DATABASE FILENAME DISK:[HOME]TEST
NUMBER OF CLUSTER NODES 1
RESERVE 6 STORAGE AREAS
RESERVE 6 JOURNALS
RESERVE 6 CACHE SLOTS
ROW CACHE IS ENABLED
CREATE STORAGE AREA RDB$SYSTEM FILENAME DISK:[HOME]TEST_SYS
CREATE STORAGE AREA TEST_A1 FILENAME DISK:[HOME]TEST_A1
CREATE STORAGE AREA TEST_A2 FILENAME DISK:[HOME]TEST_A2
CREATE CACHE TEST_A1
    CACHE SIZE 100 ROWS ROW LENGTH 100 BYTES LOCATION IS 'DISK:[HOME]'
CREATE CACHE TEST_A2
    CACHE SIZE 200 ROWS ROW LENGTH 200 BYTES LOCATION IS 'DISK:[HOME]';
DISCONNECT ALL;

ALTER DATABASE FILENAME TEST
    ADD JOURNAL TEST_J1 FILENAME DISK:[HOME]TEST_J1.AIJ
    BACKUP FILENAME DISK:[HOME]TEST_J1_BCK.AIJ
    ADD JOURNAL TEST_J2 FILENAME DISK:[HOME]TEST_J2.AIJ
    BACKUP FILENAME DISK:[HOME]TEST_J2_BCK.AIJ
    JOURNAL IS ENABLED
    (FAST COMMIT ENABLED);
%RDMS-W-DOFULLBCK, full database backup should be done to ensure future recovery
EXIT;
$
$ ! Back up the original database configuration.
$
$ RMU/BACKUP/NOLOG DISK:[HOME]TEST DISK:[HOME]TEST
$
$ ! Add new journals, row caches and storage areas
$
$ SQL
ALTER DATABASE FILENAME DISK:[HOME]TEST
    ADD JOURNAL TEST_J3 FILENAME DISK:[HOME]TEST_J3.AIJ
    BACKUP FILENAME DISK:[HOME]TEST_J3_BCK.AIJ
    ADD JOURNAL TEST_J4 FILENAME DISK:[HOME]TEST_J4.AIJ
    BACKUP FILENAME DISK:[HOME]TEST_J4_BCK.AIJ
    ADD CACHE TEST_A3
        CACHE SIZE 100 ROWS ROW LENGTH 100 BYTES LOCATION IS 'DISK:[HOME]'
    ADD CACHE TEST_A4
        CACHE SIZE 100 ROWS ROW LENGTH 100 BYTES LOCATION IS 'DISK:[HOME]'
    ADD STORAGE AREA TEST_A3 FILENAME DISK:[HOME]TEST_A3
    ADD STORAGE AREA TEST_A4 FILENAME DISK:[HOME]TEST_A4;
%RDMS-W-FULBCKREQ, The next database backup must be a full backup
EXIT;
$
$ ! Backup TEST_J1.AIJ, where the creation of the new journals, row caches
$ ! and storage areas in DISK:[HOME] is recorded.
$
$ RMU/BACKUP/AFTER/NOLOG/NOQUIET DISK:[HOME]TEST DISK:[HOME]TEST_J1_BCK.AIJ
$
$ ! Delete the database.
```


Oracle® Rdb for OpenVMS

```
$
$ SQL
  DROP DATABASE FILENAME DISK:[HOME]TEST;
  EXIT;
$
$ ! Restore the database with all storage areas, after image journals
$ ! and row caches in DISK:[HOME]
$
$ RMU/RESTORE/NOCD/NOLOG/NORECOVER/DIR=DISK:[HOME] DISK:[HOME]TEST
%RMU-I-AIJRSTAVL, 2 after-image journals available for use
%RMU-I-AIJRSTMOD, 1 after-image journal marked as "modified"
%RMU-I-AIJISON, after-image journaling has been enabled
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
$
$ ! Recover the database and change the directories from DISK:[HOME] to
$ ! DISK:[NEW] for the new storage areas, after image journals and row
$ ! caches whose creation was recorded in the backed up journal file
$ ! TEST_J1_BCK.AIJ
$
$ RMU/RECOVER/NOLOG/NOTRACE/NOCONFIRM/NOAUTOMATIC-
  /DIRECTORY=(AREAS=DISK:[NEW], -
  AFTER_JOURNAL=DISK:[NEW], -
  ROW_CACHE=DISK:[NEW]) -
  DISK:[HOME]TEST_J1_BCK.AIJ
%RMU-I-LOGRECDB, recovering database file DISK:[HOME]TEST_J1_BCK.AIJ;1
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-AIJALDONE, after-image journal roll-forward operations completed
%RMU-I-AIJSUCCESS, database recovery completed successfully
%RMU-I-AIJFNLSQ, to start another AIJ file recovery, the sequence number
needed will be 1
$
$ ! Create directory option files for all storage areas,
$ ! after image journals and row caches in DISK:[HOME]
$ ! or DISK:[NEW]
$
$ RMU/DUMP-
  /NOHEADER-
  /DIRECTORY_OPTIONS=(AREAS=DISK:[HOME]RECOVPTAREAS.OPT, -
  AFTER_JOURNAL=DISK:[HOME]RECOVPTAIJ.OPT, -
  ROW_CACHE=DISK:[HOME]RECOVPTRCACHE.OPT) -
  DISK:[HOME]TEST
$
$ ! Delete the database.
$
$ SQL$
  DROP DATABASE FILENAME DISK:[HOME]TEST;
  EXIT;
$
$ ! Restore the database with all storage areas, after image journals
$ ! and row caches again in DISK:[HOME]
$
$ RMU/RESTORE/NOCD/NOLOG/NORECOVER/DIR=DISK:[HOME] DISK:[HOME]TEST
%RMU-I-AIJRSTAVL, 2 after-image journals available for use
%RMU-I-AIJRSTMOD, 2 after-image journals marked as "modified"
%RMU-F-AIJENBOVR, enabling AIJ journaling would overwrite an existing journal
%RMU-I-AIJISOFF, after-image journaling has been disabled
$
$ ! Recover the database and again change the directories from DISK:[HOME]
$ ! to DISK:[NEW] for the new storage areas, after image journals and row
$ ! caches whose creation was recorded in the backed up journal file
$ ! TEST_J1_BCK.AIJ
$
```

Oracle® Rdb for OpenVMS

```
$ RMU/RECOVER/NOLOG/NOTRACE/NOCONFIRM/NOAUTOMATIC-
/OPTIONS=(AREAS=DISK:[HOME]RECOVPTAREAS.OPT, -
AFTER_JOURNAL=DISK:[HOME]RECOVPTAIJ.OPT, -
ROW_CACHE=DISK:[HOME]RECOVPTRCACHE.OPT) -
TEST$SCRATCH:TEST_J1_BCK.AIJ
! Recover Areas Directory Options file for database
! DISK:[HOME]TEST.RDB;1
! Created 22-JUL-2014 09:37:24.29
! Created by DUMP command

RDB$SYSTEM -
  /directory=DISK:[HOME] -
  /snapshot_directory=DISK:[HOME]

TEST_A1 -
  /directory=DISK:[HOME] -
  /snapshot_directory=DISK:[HOME]

TEST_A2 -
  /directory=DISK:[HOME] -
  /snapshot_directory=DISK:[HOME]

TEST_A3 -
  /directory=DISK:[NEW] -
  /snapshot_directory=DISK:[NEW]

TEST_A4 -
  /directory=DISK:[NEW] -
  /snapshot_directory=DISK:[NEW]
! Recover After Journal Directory Options file for database
! DISK:[HOME]TEST.RDB;1
! Created 22-JUL-2014 09:37:24.29
! Created by DUMP command

TEST_J1 -
  /directory=DISK:[HOME] -
  /backup_directory=DISK:[HOME]

TEST_J2 -
  /directory=DISK:[HOME] -
  /backup_directory=DISK:[HOME]

TEST_J3 -
  /directory=DISK:[NEW] -
  /backup_directory=DISK:[NEW]

TEST_J4 -
  /directory=DISK:[NEW] -
  /backup_directory=DISK:[NEW]
! Recover Row Cache Directory Options file for database
! DISK:[HOME]TEST.RDB;1
! Created 22-JUL-2014 09:37:24.29
! Created by DUMP command
```

```

TEST_A1 -
        /directory=DISK:[HOME]

TEST_A2 -
        /directory=DISK:[HOME]

TEST_A3 -
        /directory=DISK:[NEW]

TEST_A4 -
        /directory=DISK:[NEW]
%RMU-I-LOGRECDB, recovering database file DISK:[HOME]TEST.RDB;1
%RMU-I-AEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-AIJALDONE, after-image journal roll-forward operations completed
%RMU-I-AIJSUCCESS, database recovery completed successfully
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence number
needed will be 1
%RMU-I-AIJNOENABLD, after-image journaling has not yet been enabled

```

11.1.9 RMU Unload Record_Definition File Can Include Offset and Length Comment

The record definition (.rrd) file created by the RMU Unload Record_Definition command has been enhanced to include a comment containing each field length and offset within the output record.

This optional information is included when the qualifier /DEBUG_OPTIONS=OFFSET is included on the command line. The following example shows the comment string for each field:

```

$          RMU/UNLOAD-
          /RECORD=(FILE=SALARY_HISTORY)-
          /DEBUG_OPTIONS=OFFSET-
          PERSONNEL -
          SALARY_HISTORY -
          SALARY_HISTORY
%RMU-I-DATRECUNL, 729 data records unloaded
$          type SALARY_HISTORY.RRD
DEFINE FIELD EMPLOYEE_ID DATATYPE IS TEXT SIZE IS 5.
DEFINE FIELD SALARY_AMOUNT DATATYPE IS SIGNED LONGWORD SCALE -2.
DEFINE FIELD SALARY_START DATATYPE IS DATE.
DEFINE FIELD SALARY_END DATATYPE IS DATE.
DEFINE RECORD SALARY_HISTORY.
      EMPLOYEE_ID .                               /* Offset = 0 Length = 5 */
      SALARY_AMOUNT .                             /* Offset = 5 Length = 4 */
      SALARY_START .                              /* Offset = 9 Length = 8 */
      SALARY_END .                                /* Offset = 17 Length = 8 */
END SALARY_HISTORY RECORD.                       /* Total Length = 25 */
$

```

11.1.10 New RMU/DUMP/BACKUP Enhanced Error Handling Features

When the RMU/DUMP/BACKUP command detected a non–fatal error as it was reading an Oracle Rdb database backup file, it reported the error but continued with the dump to determine if there were other errors in the backup file. In addition, RMU/DUMP/BACKUP returned a success status in the \$STATUS symbol when it finished reading the backup file and had a normal termination, whether or not it had output errors it detected while reading the backup file.

If the RMU/DUMP/BACKUP command is being used just to verify the validity of the backup file, reading the entire backup file just to determine if it is valid can take a long time for large backup files, especially if they are on tape media. In addition, the success status in the \$STATUS symbol when the dump completed sometimes caused errors output during the dump to be missed or unnecessary time to be spent searching for any errors in an RMU/DUMP/BACKUP batch job log file.

To fix these problems, the RMU/DUMP/BACKUP error handling has been enhanced. The last most serious error detected by RMU/DUMP/BACKUP during the dump of the backup file will now always be put in the symbol \$STATUS which can be tested when RMU/DUMP/BACKUP completes or aborts by executing the VMS command "SHOW SYMBOL \$STATUS". In addition, a new [NO]EXIT_ERROR qualifier has been added to the RMU/DUMP/BACKUP command to optionally abort the dump operation as soon as an error is detected reading the backup file.

```
[NO]EXIT_ERROR
```

NOEXIT_ERROR, the default, keeps the current functionality: the RMU/DUMP/BACKUP operation will only be aborted if a fatal error is detected which prevents RMU/DUMP/BACKUP from continuing to dump the database backup file.

In the following example, the /EXIT_ERROR qualifier is specified with the RMU/DUMP/BACKUP command. When the first error is detected while reading the backup file, MF_PERSONNEL.RBF, the dump operation is aborted and the status of the error which caused the dump to be aborted, RMU–E–BLOCKLOST, is saved in the \$STATUS symbol when the RMU/DUMP/BACKUP command exits.

```
$ RMU/DUMP/BACKUP/EXIT_ERROR MF_PERSONNEL.RBF
%RMU-I-DMPTXT_163, No dump option selected. Performing read check.
%RMU-E-BLOCKLOST, block of DEVICE:[DIRECTORY]MF_PERSONNEL.RBF; lost due to
unrecoverable error
%RMU-F-FATALERR, fatal error on DUMP_BACKUP
%RMU-F-FTL_DUMP, Fatal error for DUMP operation at 1-MAY-2013 11:32:13.45
$ SHOW SYMBOL $STATUS
  $STATUS == "%X12C8821A"
```

In the following example, the /NOEXIT_ERROR qualifier is first specified with the RMU/DUMP/BACKUP command. This is the default so the second RMU/DUMP/BACKUP command, which does not specify the /NOEXIT_ERROR qualifier, has the same results as the first RMU/DUMP/BACKUP command which does specify the /NOEXIT_ERROR qualifier. When non–fatal errors are detected reading the backup file, MF_PERSONNEL.RBF, the dump operation continues and is not aborted. But now the status of the last most severe error detected reading the backup file, RMU–E–BLOCKLOST, is saved in the \$STATUS symbol when the RMU/DUMP/BACKUP command finishes reading the backup file, not a success status. A count is also given of any soft media errors which were not reported because they did not reoccur when retrying the media read operations.

```
$ RMU/DUMP/BACKUP/NOEXIT_ERROR MF_PERSONNEL.RBF
%RMU-I-DMPTXT_163, No dump option selected. Performing read check.
```

```

%RMU-E-BLOCKLOST, block of DEVICE:[DIRECTORY]MF_PERSONNEL.RBF; lost due to
unrecoverable error
%RMU-I-SOFTERRRS, 5 recoverable media errors occurred reading
DEVICE:[DIRECTORY]MF_PERSONNEL.RBF;
$ SHOW SYMBOL $STATUS
  $STATUS == "%X12C8821A"
$ RMU/DUMP/BACKUP MF_PERSONNEL.RBF
%RMU-I-DMPTXT_163, No dump option selected. Performing read check.
%RMU-E-BLOCKLOST, block of DEVICE:[DIRECTORY]MF_PERSONNEL.RBF; lost due to
unrecoverable error
%RMU-I-SOFTERRRS, 5 recoverable media errors occurred reading
DEVICE:[DIRECTORY]MF_PERSONNEL.RBF;
$ SHOW SYMBOL $STATUS
  $STATUS == "%X12C8821A"

```

11.1.11 New REVERSE Attribute for CREATE SEQUENCE Statement and IDENTITY Clause

This release of Oracle Rdb adds support for a new type of sequence. The REVERSE clause causes the value returned by NEXTVAL and CURRVAL to be bit/byte reversed. While the sequence of values computed internally by the sequence generator are regularly increasing, the values presented through the CURRVAL and NEXTVAL pseudo columns, and assigned to IDENTITY columns may not be adjacent. The advantage of such a sequence is scattered I/O when SORTED or SORTED RANKED indices are defined on such columns. This scattering of values may reduce I/O contention on nodes containing the new values generated from a normal sequence.

The new REVERSE keyword can be used in CREATE SEQUENCE, or as part of the IDENTITY clause of CREATE and ALTER TABLE statements.

The following example shows creating a table with an identity clause.

```

SQL> create table T3
cont>     (a bigint identity (reverse
cont>                               increment by 1000000
cont>                               start with -1000000)
cont>     ,rel_id integer);
SQL> insert into T3
cont>     select rel_id
cont>     from relations
cont>     order by 1 fetch
cont>     first 10 rows only;
10 rows inserted
SQL>
SQL> select t3.currval from rdb$database;

      20369552416178176
1 row selected
SQL>
SQL> table t3 order by a;
      A          REL_ID
-----
      0           2
20369552416178176 12
40739104832356352  6
81478209664712704  4
122118358450569216  8
162956419329425408  3

```

```

203279908766482432      7
244236716901138432      5
269389144898142207      1
284665759454461952      9
10 rows selected
SQL>

```

Usage Notes

- ◆ Sequences created using REVERSE generate a full 64 bit value, so columns should be created as BIGINT. Allocating a target data type that is too small will result in an *integer overflow* error as shown in the following example.

```

SQL> create table T2
cont>      (a integer identity (reverse increment by 20)
cont>      ,rel_id integer);
cont> insert into T2 select rel_id from relations;
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-COSI-F-INTOVF, integer overflow

```

- ◆ The REVERSE clause is incompatible with RANDOMIZE.
 - ◆ REVERSE sequences are maintained as a normal sequence. RDB\$NEXT_SEQUENCE_VALUE will return the current last value, but not bit/byte reversed.
-

Chapter 12

Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.1

12.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.1

12.1.1 New LIMIT_TO Qualifier Added to RMU Load Command

This release of Oracle Rdb adds a /LIMIT_TO qualifier to the RMU Load command.

The LIMIT_TO qualifier defines the maximum number of rows to read from the source data file. Depending upon the value specified for the SKIP qualifier, this also controls the number of rows written to the database.

The value of LIMIT_TO may not be zero, and the value of SKIP may not exceed this limit.

The default is NOLIMIT_TO which indicates that all rows read from the unload data file can be inserted into the database table, depending on other factors such as the value of the SKIP qualifier.

The following example shows loading a sample from the EMPLOYEES table using the LIMIT_TO qualifier.

```
$      RMU/LOAD -
      /LIMIT_TO=80 -
      /RECORD=( FILE:EMP_TXT ,FORMAT:DELIMIT) -
      PERSONNEL_SAMPLE -
      EMPLOYEES -
      EMP.TXT
DEFINE FIELD EMPLOYEE_ID DATATYPE IS TEXT SIZE IS 5.
DEFINE FIELD LAST_NAME DATATYPE IS TEXT SIZE IS 14.
DEFINE FIELD FIRST_NAME DATATYPE IS TEXT SIZE IS 10.
DEFINE FIELD MIDDLE_INITIAL DATATYPE IS TEXT SIZE IS 1.
DEFINE FIELD ADDRESS_DATA_1 DATATYPE IS TEXT SIZE IS 25.
DEFINE FIELD ADDRESS_DATA_2 DATATYPE IS TEXT SIZE IS 25.
DEFINE FIELD CITY DATATYPE IS TEXT SIZE IS 20.
DEFINE FIELD STATE DATATYPE IS TEXT SIZE IS 2.
DEFINE FIELD POSTAL_CODE DATATYPE IS TEXT SIZE IS 5.
DEFINE FIELD SEX DATATYPE IS TEXT SIZE IS 1.
DEFINE FIELD BIRTHDAY DATATYPE IS TEXT SIZE IS 16.
DEFINE FIELD STATUS_CODE DATATYPE IS TEXT SIZE IS 1.
DEFINE RECORD EMPLOYEES.
  EMPLOYEE_ID .
  LAST_NAME .
  FIRST_NAME .
  MIDDLE_INITIAL .
  ADDRESS_DATA_1 .
  ADDRESS_DATA_2 .
  CITY .
  STATE .
  POSTAL_CODE .
  SEX .
  BIRTHDAY .
  STATUS_CODE .
END EMPLOYEES RECORD.
%RMU-I-DATRECREAD, 80 data records read from input file.
```



```
%RMU-I-DATRECSTO, 80 data records stored 14-OCT-2013 07:14:03.52.
$
```

This enhancement has been added in Oracle Rdb Release 7.3.1.1.

12.1.2 New BEFORE and SINCE Qualifiers Added to RMU Load Audit

Bug 17859712

This release of Oracle Rdb adds new qualifiers to RMU Load Audit to allow audit records to be filtered by timestamp. The qualifiers BEFORE and SINCE can specify the date/time range which will be extracted from the OpenVMS audit journal and saved in the target auditing table.

These qualifiers accept the standard OpenVMS date/time specification that includes special keywords such as YESTERDAY, TODAY and TOMORROW. These values can be very effective when used with the List_Plan qualifier and used later when using RMU Load Plan.

If these qualifiers are omitted, then their values default to minimum and maximum possible date/time values.

This example shows the use of DCL symbols to be used at runtime to provide the date/time range.

```
$ RMU/LOAD/AUDIT -
  /SINCE=&start_ts -
  /BEFORE=&end_ts -
  TESTDB AUDIT_RECORDS -
  SYS$COMMON:[SYSMGR]SECURITY.AUDIT$JOURNAL
%RMU-I-DATRECREAD, 91 data records read from input file.
%RMU-I-DATRECSTO, 63 data records stored 27-NOV-2013 00:59:33.18.
$
```

This example uses explicit date and time values to load a specific range of audit records.

```
$ RMU/LOAD/AUDIT -
  /SINCE=1-JAN-2011 -
  /BEFORE="1-NOV-2013 13:00" -
  TESTDB AUDIT_RECORDS -
  SYS$COMMON:[SYSMGR]SECURITY.AUDIT$JOURNAL
%RMU-I-DATRECREAD, 91 data records read from input file.
%RMU-I-DATRECSTO, 0 data records stored 27-NOV-2013 00:59:33.75.
$
```

Additionally, RMU/LOAD/PLAN now supports the AUDIT keyword and the new associated BEFORE and SINCE keywords that correspond to this new functionality.

```
$ RMU/LOAD/AUDIT -
  /SINCE=YESTERDAY -
  /NOEXECUTE -
  /LIST_PLAN=SAMPLE.PLAN -
  TESTDB AUDIT_RECORDS -
  SYS$COMMON:[SYSMGR]SECURITY.AUDIT$JOURNAL
$
```

Later the Plan can be executed and inherit the current value for YESTERDAY.

```

$ RMU/LOAD/PLAN SAMPLE.PLAN
%RMU-I-PROCPLNFIL, Processing plan file SAMPLE.PLAN.
  ! Plan created on 28-NOV-2013 by RMU/LOAD.

Plan Name = LOAD_PLAN
Plan Type = LOAD

Plan Parameters:
  Database Root File = DISK1:[TESTING]TESTDB.RDB;
  Table Name = AUDIT_RECORDS
  Input File = SYS$COMMON:[SYSMGR]SECURITY.AUDIT$JOURNAL
  Audit = TESTDB
    Since = "YESTERDAY"

  ! Fields = <all>
  NoVirtual_Fields
  NoMatch_Name
  Dialect = SQL99
  Transaction_Type = PROTECTED
  ! Buffers = <default>
  ! Commit_Every = <never>
  Row_Count = 500
  ! Skip = <none>
  ! Limit_To = <none>
  NoReplace_Rows
  NoLog_Commits
  NoCorresponding
  NoDefer_Index_Updates
  Constraints
  NoParallel
  NoRestricted_Access
  NoPlace
  ! Statistics = <none>
  ! Trigger_Relations = <not specified>
End Plan Parameters

Executor Parameters:
  Executor Name = EXECUTOR_1
  ! Place_Only = <none>
  ! Exception_File = <none>
  ! RUJ Directory = <default>
  Communication Buffers = 1
End Executor Parameters
%RMU-I-DATRECREAD, 195 data records read from input file.
%RMU-I-DATRECSTO, 21 data records stored 28-NOV-2013 23:34:50.27.
$

```

This enhancement has been added in Oracle Rdb Release 7.3.1.1.

12.1.3 New RMU/SHOW/STATISTICS Output File Periodic Buffer Flushes

When a system failure occurred, important diagnostic data could be lost from the Oracle Rdb RMU/SHOW STATISTICS output files: the binary file used to record Oracle Rdb database statistics

for later replay; the logical area access log file; the record access dbkey log file; the process deadlock log file; the lock timeout log file; the OPCOM messages log file; the Hot Standby log file; the online analysis log file; and the stall messages log file. To minimize the loss of diagnostic data from these RMU/SHOW STATISTICS output files, periodic buffer data flushes will now occur if these files are created. These periodic output file data flushes will be the default. The user will be able to modify the periodic flush interval or specify that periodic buffer flushes are not to occur.

The syntax for this new RMU/SHOW STATISTICS qualifier is as follows.

```
/FLUSH_INTERVAL=seconds
/NOFLUSH_INTERVAL
```

The default if the new /FLUSH_INTERVAL qualifier is not specified will be a periodic flush interval of 60 seconds. The minimum flush interval that can be specified is 0 seconds. The maximum flush interval that can be specified is 3600 seconds (1 hour). Specifying 0 seconds for the flush interval is equivalent to specifying /NOFLUSH_INTERVAL. If the flush interval is active and less than the statistics collection interval, to avoid unnecessary buffer flushes the flush interval will be set to the statistics collection interval, which has a default of 3 seconds and can be set by the existing /TIME qualifier, or by typing an "S" if "Set rate" is displayed at the bottom of the statistics screen. The current collection interval is displayed following "Rate:" in the statistics screen header.

The first and second command examples below use the default flush interval of 60 seconds. The third and fourth command examples below specify a flush interval of 10 seconds. Note that the flush interval can be set even if the RMU/SHOW STATISTICS command does not specify any log or output files. This is because the TOOLS menu can be used later in the RMU/SHOW STATISTICS session to create log files for which the flush interval of 60 or 10 will be used.

```
$ RMU/SHOW STATISTICS MF_PERSONNEL
$ RMU/SHOW STATISTICS/OUTPUT=SHOWSTAT.DAT/DBKEY_LOG=D.LOG MF_PERSONNEL
$ RMU/SHOW STATISTICS/FLUSH_INTERVAL=10 MF_PERSONNEL
$ RMU/SHOW STATISTICS/FLUSH_INTERVAL=10/OUTPUT=SHOWSTAT.DAT/DBKEY_LOG=D.LOG
MF_PERSONNEL
```

The following command example specifies a statistics collection interval of 12 seconds using the /TIME command qualifier. This is larger than the 10 seconds specified by the new /FLUSH_INTERVAL qualifier. Since the collection interval is larger than the flush interval, the collection interval will be used for the flush interval to avoid excess buffer flushes.

```
$ RMU/SHOW STATISTICS/FLUSH_INTERVAL=10/TIME=12/OUTPUT=SHOWSTAT.DAT
/DBKEY_LOG=D.LOG MF_PERSONNEL
```

The following command examples are equivalent and specify that periodic buffer flushes will not be used for the RMU/SHOW STATISTICS binary output and log files.

```
$ RMU/SHOW STATISTICS/NOFLUSH_INTERVAL/OUTPUT=SHOWSTAT.DAT/DBKEY_LOG=D.LOG
MF_PERSONNEL
$ RMU/SHOW STATISTICS/FLUSH_INTERVAL=0/OUTPUT=SHOWSTAT.DAT/DBKEY_LOG=D.LOG
MF_PERSONNEL
```

This enhancement has been added to Oracle Rdb Release 7.3.1.1.

12.1.4 New Error and Log Messages Added for Segmented String Verification

To verify segmented strings for all Oracle Rdb database tables, the commands `RMU/VERIFY/ALL` or `RMU/VERIFY/SEGMENTED_STRINGS/LAREAS` or `RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=*` can be used. To verify segmented strings for individual tables, the `/LAREAS` qualifier must be used with the `/SEGMENTED_STRINGS` qualifier to specify one or more names of tables to be verified.

There was a problem where, if the `/LAREAS` qualifier was used with the `/SEGMENTED_STRINGS` qualifier, logical area identifier numbers or the names of logical areas that were not tables could be specified with the `/LAREAS` qualifier without an error, even though segmented string data is verified only on a table-wide basis for a table which contains segmented string columns, including all table records that may be vertically or horizontally partitioned across multiple logical areas.

Allowing logical area id numbers or the names of logical areas that were not table names to be specified could cause confusion or lead the user to falsely assume that segmented strings contained in these logical areas were being verified. Therefore, the `RMU/VERIFY` operation will now be aborted and a new fatal `"%RMU-I-TBLSEGVER"` error will be output if the `/SEGMENTED_STRINGS` qualifier is specified in the same `RMU/VERIFY` command as the `/LAREAS` qualifier and the `/LAREAS` qualifier specifies a logical area id or a logical area name which is not a valid table name.

The following example shows the previous behavior. In the first command, "RESUMES" is a valid table that contains segmented strings and the segmented strings are therefore verified. In the second command, `RMU/VERIFY` detected that the "NOTATABLE" table did not exist and returned the fatal `"%RMU-F-NOTLAREA"` error. In the third command, the logical area id number "95" was ignored and no segmented string verification took place but the user could wrongly assume that segmented string data had been verified. In the fourth command, the index logical area name "SH_EMPLOYEE_ID" was also ignored so the user could again wrongly assume that segmented string data had been verified.

```
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=RESUMES MF_PERSONNEL
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=NOTATABLE MF_PERSONNEL
%RMU-F-NOTLAREA, "NOTATABLE" is not a valid logical area name or number
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=95 MF_PERSONNEL
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=SH_EMPLOYEE_ID MF_PERSONNEL
```

The following example shows the new behavior. In the first command, "RESUMES" is a valid table that contains segmented strings so the segmented string data is verified as previously. In the second command, a fatal error is returned as previously but the error is detected earlier and a more specific error message is output using the new `"%RMU-F-INVSEGTBL"` fatal error message. In the third command and the fourth command, the invalid table names are now detected and the new `"%RMU-F-INVSEGTBL"` fatal error message is output.

```
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=RESUMES MF_PERSONNEL
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=NOTATABLE MF_PERSONNEL
%RMU-F-INVSEGTBL, Invalid table name NOTATABLE specified for segmented string
verification
%RMU-F-FTL_VER, Fatal error for VERIFY operation at 27-JAN-2014 15:09:05.08
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=95 MF_PERSONNEL
%RMU-F-INVSEGTBL, Invalid table name 95 specified for segmented string
verification
```

Oracle® Rdb for OpenVMS

```
%RMU-F-FTL_VER, Fatal error for VERIFY operation at 27-JAN-2014 15:02:39.04
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=SH_EMPLOYEE_ID MF_PERSONNEL
%RMU-F-INVSEGTBL, Invalid table name SH_EMPLOYEE_ID specified for segmented
string verification
%RMU-F-FTL_VER, Fatal error for VERIFY operation at 27-JAN-2014 15:03:42.15
```

A new LOG message has also been added to RMU/VERIFY to list the tables containing columns defined for segmented string data for which the segmented string data will be verified. If log messages are activated for the RMU/VERIFY command, the new log message "%RMU-I-TBLSEGVER" will be output at the beginning of the verify operation and will be repeated for each table selected for segmented string data verification. In the first command, only the "RESUMES" table specified by the /LAREA qualifier is verified. In the second command, "RMU/VERIFY/ALL" is specified which, by default, verifies all tables in the database with segmented string columns, including the system tables. Note that in the second command, most of the log messages that follow the "%RMU-I-TBLSEGVER" messages have been left out to save space.

```
$ RMU/VERIFY/LOG/SEGMENTED_STRINGS/LAREAS=RESUMES MF_PERSONNEL
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RESUMES
%RMU-I-DBBOUND, bound to database "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1"
%RMU-I-OPENAREA, opened storage area RDB$SYSTEM for protected retrieval
%RMU-I-BGNAIPVER, beginning AIP pages verification
%RMU-I-ENDAIPVER, completed AIP pages verification
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-I-OPENAREA, opened storage area MF_PERS_SEGSTR for protected retrieval
%RMU-I-ENDABMSPM, completed ABM pages verification
%RMU-I-BSGPGLARE, beginning verification of RESUMES logical area
as part of EMP_INFO storage area
%RMU-I-OPENAREA, opened storage area EMP_INFO for protected retrieval
%RMU-I-ESGPGLARE, completed verification of RESUMES logical area
as part of EMP_INFO storage area
%RMU-I-CLOSAREAS, releasing protected retrieval lock on all storage areas
%RMU-S-ENDVERIFY, elapsed time for verification : 0 00:00:00.80
$ RMU/VERIFY/ALL/LOG MF_PERSONNEL
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-TBLSEGVER, Segmented strings will be verified for table CANDIDATES
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RESUMES
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RDB$COLLATIONS
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RDB$CONSTRAINTS
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RDB$DATABASE
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RDB$FIELDS
%RMU-I-TBLSEGVER, Segmented strings will be verified for table
RDB$FIELD_VERSIONS
%RMU-I-TBLSEGVER, Segmented strings will be verified for table
RDB$INDEX_SEGMENTS
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RDB$INDICES
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RDB$MODULES
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RDB$PARAMETERS
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RDB$PRIVILEGES
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RDB$PROFILES
%RMU-I-TBLSEGVER, Segmented strings will be verified for table
RDB$QUERY_OUTLINES
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RDB$RELATIONS
%RMU-I-TBLSEGVER, Segmented strings will be verified for table
RDB$RELATION_CONSTRAINTS
%RMU-I-TBLSEGVER, Segmented strings will be verified for table
RDB$RELATION_FIELDS
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RDB$ROUTINES
```

Oracle® Rdb for OpenVMS

```
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RDB$SEQUENCES
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RDB$STORAGE_MAPS
%RMU-I-TBLSEGVER, Segmented strings will be verified for table
RDB$STORAGE_MAP_AREAS
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RDB$TRIGGERS
%RMU-I-TBLSEGVER, Segmented strings will be verified for table
RDB$TRIGGER_ACTIONS
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RDB$TYPES
%RMU-I-TBLSEGVER, Segmented strings will be verified for table RDB$TYPE_FIELDS
%RMU-I-BGNVCONST, beginning verification of constraints for database
DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
%RMU-I-ENDVCONST, completed verification of constraints for database
DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
%RMU-I-DBBOUND, bound to database "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1"
$
```

Chapter 13

Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.0

13.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.0

13.1.1 Changes to Default and Limits Behavior in Oracle Rdb

This release of Oracle Rdb changes the following default behavior.

CREATE DATABASE Statement

These new defaults will be used when creating a database.

- ◆ The default PAGE SIZE changes from 2 to 4 blocks
- ◆ The default BUFFER SIZE changes from 3 pages to 4 pages
- ◆ The default NUMBER OF BUFFERS changes from 20 to 250 buffers
- ◆ The default NUMBER OF RECOVERY BUFFERS changes from 20 to 250 buffers
- ◆ The default for SECURITY CHECKING clause is now (PERSONA IS ENABLED)
- ◆ The default for SYSTEM INDEX is now (TYPE IS SORTED RANKED, COMPRESSION IS ENABLED)

The SYSTEM INDEX changes are applied to all new databases created with CREATE DATABASE statement and IMPORT DATABASE statement. Older databases converted using RMU/CONVERT or RMU/RESTORE (with the implicit Convert action) will also result in the new SYSTEM INDEX default.

The other new defaults do not affect databases created in Rdb V7.2 (or older versions) that are converted to Oracle Rdb V7.3.1 (or later) using RMU/CONVERT or RMU/RESTORE. In most cases, recreating databases using the SQL IMPORT DATABASE statement using an interchange file (.rbr) from older versions of Oracle Rdb will preserve the settings from the source database.

Interchanges files (.rbr) created with Oracle Rdb SQL EXPORT from V7.2.4 and later do export the PAGE SIZE of the database. However, older versions of Rdb did not export the PAGE SIZE if it was 2 pages (the old default). If you are using older interchange files then the IMPORT DATABASE statement should include PAGE SIZE definitions for the database and each storage area that used PAGE SIZE 2. Tools such as RMU/EXTRACT/ITEM=IMPORT can be used to create a script for this purpose.

These new limits are now enforced by Oracle Rdb.

- ◆ The maximum BUFFER SIZE can now be specified up to 256 blocks. Previously, the maximum allowed database buffer size was 128 blocks. Be aware that using larger database buffer sizes will require additional virtual memory.
- ◆ The minimum NUMBER OF USERS is now 5. In prior versions of Oracle Rdb, the minimum number of allowed database users was one (1). This minimum has been increased to five to allow for various optional database servers (such as the ABS or RCS or ALS) to access the database. When RMU Convert or SQL IMPORT DATABASE are used to create databases in Rdb Release 7.3, they will automatically establish a new minimum if the one defined for the

original database was less than 5 users.

ALTER DATABASE Statement

When adding a new storage area to a database, that new storage area will assume a default PAGE SIZE of 4 blocks. This may be problematic if the database has a small (for instance the default) BUFFER SIZE from older database versions.

In this example, a database created under Oracle Rdb Release 7.2.5.3 was converted to Rdb Release 7.3.1.0.

```
SQL> alter database filename ABC add storage area XYZ;
SQL> attach 'filename ABC';
SQL> show storage area XYZ
```

```

XYZ
  Access is:      Read write
  Page Format:    Uniform
  Page Size:     4 blocks
  Area File:     USER2:[TESTING]XYZ.RDA;1
  Area Allocation: 700 pages
  Extent:        Enabled
  Area Extent Minimum: 99 pages
  Area Extent Maximum: 9999 pages
  Area Extent Percent: 20 percent
  Snapshot File: USER2:[TESTING]XYZ.SNP;1
  Snapshot Allocation: 100 pages
  Snapshot Extent Minimum: 99 pages
  Snapshot Extent Maximum: 9999 pages
  Snapshot Extent Percent: 20 percent
  Locking is Row Level
  No Cache Associated with Storage Area
No database objects use Storage Area XYZ
SQL>
```

The problem lies in the fact that the current BUFFER SIZE is only 6 blocks (see the SHOW DATABASE output below). This would mean that I/O to the new storage area would only be adding one page to the buffer, with over 33% of the buffer wasted.

```
SQL> show database rdb$dbhandle
Default alias:
  Oracle Rdb database in file ABC
  Multischema mode is disabled
  Number of users: 50
  Number of nodes: 16
  Buffer Size (blocks/buffer): 6
  Number of Buffers: 20
  Number of Recovery Buffers: 20
.
.
.
```

Oracle recommends that an explicit PAGE SIZE clause be used when defining a new storage area.

CREATE INDEX Statement

These new defaults will be used when creating an index.

- ◆ The default NODE SIZE for SORTED RANKED and unique SORTED indices is now chosen to be large enough to fill the free space in the new logical area. In prior versions, a smaller NODE SIZE was chosen based on the index key length. However, these small nodes left unused space on the database pages and so were wasteful of disk space and virtual memory (when in buffers).
No changes were made to the default node size for SORTED indices with duplicates. In this case, the smaller duplicate nodes may fill the unused space on the page.
- ◆ The default PERCENT FILL changes from 70% to 85%

These changes do not affect indices created in Rdb V7.2 (or older versions) when the database is converted to Oracle Rdb Release 7.3.1 using RMU/CONVERT or RMU/RESTORE.

Importing a database using the IMPORT DATABASE statement will fix up the metadata for the PERCENT FILL and NODE SIZE so that the output from SHOW INDEX will provide more information than in prior versions.

Logical Name RDMS\$BIND_WORK_VM

The Oracle Rdb query optimizer might make use of temporary virtual memory (VM) during query processing. This memory is used to cache index keys during zig-zag match strategy and dbkey lists during temporary-relation processing (not related to the SQL temporary table feature). In either case, the virtual memory size defined by the logical name RDMS\$BIND_WORK_VM is used for each buffer which might overflow to a temporary disk file (located using the RDMS\$BIND_WORK_FILE logical name).

With this release the new default has increased from 10,000 to 100,000 bytes. This change makes it more likely that queries can complete entirely in VM rather than opening and using a small disk file.

Logical Name RDMS\$BIND_MAX_QSORT_COUNT

When the number of rows is relatively small, the Oracle Rdb query processor can avoid using SORT32 (which has a higher setup cost) by using an in-memory Quick Sort. In prior versions, the default threshold was 63 rows. This release of Oracle Rdb defaults to 5,000 rows. The larger threshold should allow more sorting to consume less resources. This threshold can be changed (for instance to return to the prior default of 63) using the logical name RDMS\$BIND_MAX_QSORT_COUNT.

13.1.2 New /ERROR_LIMIT Qualifier Added as the Default to RMU/VERIFY

New default functionality has been added to RMU/VERIFY to limit the number of diagnostics output when verifying an Oracle Rdb database. By default, RMU/VERIFY will now limit the number of diagnostic messages output by RMU/VERIFY to 100. If this limit is exceeded, the RMU/VERIFY will terminate with a warning message.

```
%RMU-W-MAXVERERR, Maximum error limit 100 exceeded - ending verification
```

To disable this behavior and have no limit on the number of diagnostic messages output by the verification, /NOERROR_LIMIT must be specified on the command line.

```
RMU/VERIFY/ALL/NOERROR_LIMIT mf_personnel
```

Oracle® Rdb for OpenVMS

To override the default of 100, the user can specify a numeric value for the /ERROR_LIMIT between 1 and 2147483647.

```
RMU/VERIFY/ALL/ERROR_LIMIT=50 mf_personnel
```

If /ERROR_LIMIT is specified without a value, the default limit of 100 diagnostics will be used.

```
RMU/VERIFY/ALL/ERROR_LIMIT mf_personnel
```

The /ERROR_LIMIT does not include any logging messages put out when the /LOG qualifier is used with RMU/VERIFY. It only includes diagnostic messages which are defined as messages of any severity which are not logging messages put out when the /LOG qualifier is specified. The %RMU-W-MAXVERERR message is not included in the /ERROR_LIMIT count but is output once the /ERROR_LIMIT in force is exceeded in place of the diagnostic message that would have exceeded the error limit.

We have done everything we can to make the /ERROR_LIMIT count as accurate as possible but related messages output together in one output operation may be counted as one message in a limited number of cases. Therefore, we do not guarantee absolute accuracy in the /ERROR_LIMIT count in all cases but consider it as an acceptably accurate way to limit the potentially large number of diagnostics that can be output by the RMU/VERIFY of a database.

The syntax for this qualifier is as follows:

```
/[NO]ERROR_LIMIT[=n]
```

"n" is a positive numeric value between 1 and 2147483647.

In the following example, the RMU/VERIFY of a database completes normally with 6 diagnostics since the default error limit of 100 is not exceeded.

```
$ rmu/verify/all mf_personnel
%RMU-W-PAGCKSBAD, area EMP_INFO, page 2
    contains an invalid checksum
    expected: A77B3D6D, found: A7713D6D
%RMU-W-PAGLIXFRS, area EMP_INFO, page 2
    line index entry 15 maps free space at offset 00000106 (hex)
%RMU-W-PAGLIXSML, area EMP_INFO, page 2
    line index entry 19, length too small
    expected at least 2, found: 0
%RMU-W-PAGLIXSML, area EMP_INFO, page 2
    line index entry 20, length too small
    expected at least 2, found: 0
%RMU-W-PAGLIXSML, area EMP_INFO, page 2
    line index entry 21, length too small
    expected at least 2, found: 0
%RMU-W-PAGLIXSML, area EMP_INFO, page 2
    line index entry 22, length too small
    expected at least 2, found: 0
$
```

In the following example, the RMU/VERIFY is of the same database as in the previous example but an error limit of 5 diagnostic messages is specified. Therefore, 5 diagnostic messages are output and instead of the 6th diagnostic message being output, the %RMU-W-MAXVERERR is output and the database verify terminates.

Oracle® Rdb for OpenVMS

```
$ rmu/verify/all/error_limit=5 mf_personnel
%RMU-W-PAGCKSBAD, area EMP_INFO, page 2
    contains an invalid checksum
    expected: A77B3D6D, found: A7713D6D
%RMU-W-PAGLIXFRS, area EMP_INFO, page 2
    line index entry 15 maps free space at offset 00000106 (hex)
%RMU-W-PAGLIXSML, area EMP_INFO, page 2
    line index entry 19, length too small
    expected at least 2, found: 0
%RMU-W-PAGLIXSML, area EMP_INFO, page 2
    line index entry 20, length too small
    expected at least 2, found: 0
%RMU-W-PAGLIXSML, area EMP_INFO, page 2
    line index entry 21, length too small
    expected at least 2, found: 0
%RMU-W-MAXVERERR, Maxium error limit 5 exceeded - ending verification
```

13.1.3 RMU /VERIFY Root Displays the Corrupt Page Table Entries

Bug 870984

In previous releases of Oracle Rdb, the command RMU/VERIFY ROOT did not show any Corrupt Page Table (CPT) entries even when they existed. The commands RMU/SHOW CORRUPT, RMU/DUMP/HEAD=CORRUPT and RMU/VERIFY/ALL would show them.

A new feature has been added to RMU/VERIFY ROOT so that it now displays the CPT entries.

The following example shows that RMU/VERIFY/ROOT now displays the corrupt page entries in the database corrupt page table. The RMU/SHOW CORRUPT command is first executed to display the corrupt page entries in the corrupt page table for the PERSONNEL database. When the RMU/VERIFY/ROOT command is then executed it outputs a message for each corrupt page entry in the corrupt page table. This is the same message output by the RMU/VERIFY/ALL command.

```
$ RMU/SHOW CORRUPT PERSONNEL
*-----*
* Oracle Rdb V7.3-100                                     1-FEB-2013 16:41:40.77
*
* Dump of Corrupt Page Table
*   Database: DEVICE:[DIRECTORY]PERSONNEL.RDB;2
*
*-----*
Entries for storage area RDB$SYSTEM
-----

    Page 300
    - AIJ recovery sequence number is -1
    - Live area ID number is 1
    - Consistency transaction sequence number is 0
    - State of page is: corrupt

*-----*
* Oracle Rdb V7.3-100                                     1-FEB-2013 16:41:40.77
*
* Dump of Storage Area State Information
```

```
* Database: DEVICE:[DIRECTORY]PERSONNEL.RDB;2
*
*-----
```

All storage areas are consistent.

```
$ RMU/VERIFY/ROOT/LOG PERSONNEL
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-DBBOUND, bound to database "DEVICE:[DIRECTORY]PERSONNEL.RDB;2"
%RMU-I-OPENAREA, opened storage area DEVICE:[DIRECTORY]PERSONNEL.RDB;2 for protected retr
%RMU-I-BGNAIPVER, beginning AIP pages verification
%RMU-I-ENDAIPVER, completed AIP pages verification
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-I-ENDABMSPM, completed ABM pages verification
%RMU-E-CORRUPTPG, Page 300 in area DEVICE:[DIRECTORY]PERSONNEL.RDB;2 is marked as corrupt.
%RMU-I-CLOSAREAS, releasing protected retrieval lock on all storage areas
%RMU-S-ENDVERIFY, elapsed time for verification : 0 00:00:00.49
$
```

13.1.4 DECLARE LOCAL TEMPORARY TABLE Supports COMMENT IS Clause

With this release of Oracle Rdb, the DECLARE LOCAL TEMPORARY TABLE statement now supports the COMMENT IS clause. This comment is not stored by Rdb but can be used to document the DECLARE statement when it appears in a CREATE MODULE statement or when used in Interactive SQL scripts.

The following example shows the placement of the clause.

```
SQL> declare local temporary table module.STBL
cont>      (a int)
cont>      on commit preserve rows
cont>      comment is 'Test for local temporary table'
cont>      large memory is enabled
cont> ;
```

For further details please refer to the Oracle Rdb SQL Reference Manual.

13.1.5 Temporary Tables Now Support LARGE MEMORY Option

With this release of Oracle Rdb, the DECLARE LOCAL TEMPORARY TABLE statement, the CREATE GLOBAL TEMPORARY TABLE statement, and the CREATE LOCAL TEMPORARY TABLE statement all support the use of LARGE MEMORY on OpenVMS.

A new LARGE MEMORY IS { ENABLED | DISABLED } clause has been added to these statements so that the temporary table virtual memory now resides in 64 bit memory. This allows much larger temporary tables than in previous releases of Oracle Rdb.

The following example shows the placement of the clause.

```
SQL> create local temporary table LTBL
cont>     (a int)
cont>     on commit preserve rows
cont>     comment is 'Test for local temporary table'
cont>     large memory is enabled
cont> ;
SQL> show table (column) LTBL;
Information for table LTBL

A local temporary table.
On commit Preserve rows
      Large Memory:           Enabled

Columns for table LTBL:
Column Name                   Data Type           Domain
-----
A                               INTEGER
```

SQL>

Additionally, the ALTER TABLE statement has been enhanced to enable (or disable) this feature on existing temporary table definitions. This clause is not permitted for information on base tables.

For further details please refer to the Oracle SQL Reference Manual.

13.1.6 COUNT Now Returns BIGINT Result

The SQL aggregate function COUNT now returns BIGINT (aka 64 bit values) in this release of Oracle Rdb (Release 7.3.1.0). This change has been made to accommodate large tables and result sets.

The side effects of this change that may be visible are:

- ◆ Wider column output in interactive SQL for queries that perform SELECT COUNT(*), COUNT(DISTINCT expression) and COUNT(expression).
- ◆ COMPUTED BY and AUTOMATIC columns will now be displayed as BIGINT type because of an expression that uses a COUNT function.
- ◆ SQLDA data type change for COUNT expressions.

In general, this change is backward compatible with existing applications. The computed BIGINT value will automatically be converted to INTEGER for older SQL precompiler or SQL module language applications.

13.1.7 Aggregate Functions Now Use BIGINT Counters

COUNT and related aggregate functions AVG, VARIANCE, and STDDEV all process counters in BIGINT registers. This allows Rdb to process aggregation across much larger row sets than in previous releases.

With this release of Oracle Rdb, the COUNT aggregate function will return BIGINT data type results. In prior releases, an INTEGER type was the result. Applications that create COMPUTED BY or AUTOMATIC columns may notice that the data type of such columns changed to BIGINT.

Note

Oracle Rdb will implicitly convert internal results to INTEGER if the target data type in the application has not changed.

13.1.8 /[NO]KEY_VALUES Qualifier Added to RMU/VERIFY/INDEX

A new /KEY_VALUES qualifier has been added to the Oracle Rdb RMU/VERIFY command for verifying the integrity of Rdb databases. The /KEY_VALUES qualifier verifies the key field values contained in a sorted, sorted ranked or hashed index against the key field values in the matching table row to make sure the key field values contained in the index match the field values in the table. This qualifier can only be specified if indexes are being verified explicitly as with the RMU/VERIFY/INDEX command or if indexes are being verified by default such as with the RMU/VERIFY/ALL command. Key field values will be verified for all indexes or a specified list of one or more indexes. Diagnostic error messages will be put out if the index key field value does not match the table row key field value, if a DBKEY contained in the index node or hash bucket does not match the DBKEY of a table row, or if the DBKEY of a table row is not contained in the index node or hash bucket. Indexes with one key field or multiple key fields can be verified using the /KEY_VALUES qualifier.

The /KEY_VALUES qualifier will ignore index fields that have a collating sequence or are of type character varying. The character encoding for the collating sequence prevents a byte by byte comparison with the column row values. The key encoding for character varying pads with spaces so the precise length cannot be compared with the column row values. For these index fields, a warning message will be output and a comparison with the column row values will not be done.

The syntax for this new qualifier is as follows –

```
/[NO]KEY_VALUES
```

Note that /KEY_VALUES is not the default and must be specified.

The following example shows the error message put out if the index key field value in the index structure does not match the key field value in the table row. The DBKEY pointer to the index node or hash bucket that contains the row DBKEY is output as well as the table row DBKEY.

```
$ RMU/VERIFY/INDEX=JH_EMPLOYEE_ID/KEY_VALUES MF_PERSONNEL
%RMU-E-BADIDXFLD, Index JH_EMPLOYEE_ID key field EMPLOYEE_ID value at dbkey
93:810:3 does not match stored value for table JOB_HISTORY at dbkey 90:289:5 .
```

The following example shows the error message put out if the key field value(s) returned from index only retrieval of the key fields do not match on row DBKEY with the key fields returned sequentially from the table row. Because the retrieval is out of sequence, the key field values cannot be compared.

```
$ RMU/VERIFY/INDEX/KEY_VALUES MF_PERSONNEL
%RMU-E-BADIDXDBK, Index JH_EMPLOYEE_ID dbkey 91:413:3 does not match
table JOB_HISTORY row dbkey 90:200:4 .
```

The following example shows the error message put out if the table contains a DBKEY pointing to a table row that is not in the index.

Oracle® Rdb for OpenVMS

```
$ RMU/VERIFY/INDEX/KEY_VALUES/ERROR_LIMIT=200 MF_PERSONNEL
%RMU-E-NOTIDXDBK, Table COLLEGES row dbkey 68:2:1 is not in index
COLL_COLLEGE_CODE .
```

The following example shows the error message put out if the index contains a DBKEY pointing to a table row that is not in the table.

```
$ RMU/VERIFY/INDEX/KEY_VALUES/ERROR=200 MF_PERSONNEL
%RMU-E-NOTTABDBK, Index COLL_COLLEGE_CODE dbkey 68:2:1 is not in table
COLLEGES .
```

The following example shows that a warning message is put out for index key fields that have a collating sequence or that are of type character varying. These index fields cannot be compared with row values using the /KEY_VALUES qualifier.

```
$ RMU/VERIFY/INDEX/KEY_VALUES ABC.RDB
%RMU-W-NOTTYPNDX, Index MANUFACTURING_INDEX field MANUFACTURER_NAME cannot
be verified with the row value in table MANUFACTURING;
reason - COLLATING SEQUENCE.
$ RMU/VERIFY/INDEX/KEY_VALUES DBASE_INDEX.RDB
%RMU-W-NOTTYPNDX, Index FORECAST_HASH field PART_NO cannot be verified with
the row value in table FORECAST_VOLUME; reason - CHARACTER VARYING.
```

13.1.9 The /LOCK_TIMEOUT Qualifier Now Allows the Database Default

For the Oracle Rdb RMU commands RMU/BACKUP/ONLINE, RMU/COPY/ONLINE, and RMU/BACKUP/AFTER/QUIET_POINT, the /LOCK_TIMEOUT qualifier can be specified. Previously the /LOCK_TIMEOUT qualifier required a value, the maximum time in seconds to wait for acquiring the database QUIET POINT and other locks used for online database access. If "/LOCK_TIMEOUT = n" was not specified, RMU would wait indefinitely to acquire the database lock it needed.

Now the /LOCK_TIMEOUT qualifier can be specified without a value. In this case, the default lock timeout value specified for the database will be used. Specifically, the default lock timeout value used will be the value of the logical name RDM\$BIND_LOCK_TIMEOUT_INTERVAL if it has been specified, otherwise the "LOCK TIMEOUT INTERVAL" specified by the SQL CREATE DATABASE or ALTER DATABASE command will be used. If neither value has been specified, the value used will be the maximum possible lock timeout value which can be specified for an Oracle Rdb database.

The new syntax for this qualifier is as follows –

```
/LOCK_TIMEOUT [ = n ]
```

Note that /LOCK_TIMEOUT is not the default and must be specified. The default, if /LOCK_TIMEOUT is not specified, continues to be to wait indefinitely to acquire the QUIET POINT or other database locks requested by RMU.

The following example shows the different RMU commands which accept the /LOCK_TIMEOUT qualifier. In the first command, the specified lock timeout value of 100 seconds will be used. In the

other commands, since a lock timeout value is not specified, the default database lock timeout value described above will now be used.

```
$ RMU/BACKUP/ONLINE/LOCK_TIMEOUT=100/NOLOG MF_PERSONNEL MFP.RBF
$ RMU/BACKUP/ONLINE/LOCK_TIMEOUT/NOLOG MF_PERSONNEL MFP.RBF
$ RMU/COPY/ONLINE/LOCK_TIMEOUT/ROOT=DISK:[DIRECTORY]MF_PERSONNEL-
  /NOAFTER/NOLOG MF_PERSONNEL
$ RMU/BACKUP/AFTER/NOLOG/QUIET_POINT/LOCK_TIMEOUT -
  DISK:[DIRECTORY]MF_PERSONNEL MFP_AIJ_1
```

The following example shows that the PLAN file used with the RMU PARALLEL BACKUP command will accept either the specified lock timeout value of 100 seconds or will now accept the default database lock timeout value described above.

```
$ RMU/BACKUP/PARALLEL=EXEC=1/DISK=WRITER=2/LIST=TMP.PLAN/ONLINE-
  /LOCK_TIMEOUT=100 MF_PERSONNEL DISK:[DIRECTORY]MFP,DISK:[DIRECTORY]
$ SEAR TMP.PLAN LOCK_TIMEOUT
  Lock_Timeout = 100
$ RMU/BACKUP/PARALLEL=EXEC=1/DISK=WRITER=2/LIST=TMP.PLAN/ONLINE-
  /LOCK_TIMEOUT MF_PERSONNEL DISK:[DIRECTORY]MFP,DISK:[DIRECTORY]
$ SEAR TMP.PLAN LOCK_TIMEOUT
  Lock_Timeout
$ RMU/BACKUP/PLAN TMP.PLAN
```

13.1.10 Compression of AIJ Backup Files for Automatic AIJ Backups

A new feature has been added to allow compression of AIJ files during automatic AIJ backups.

Also backups of after image journals using /Format=Old_File can be compressed using the ZLIB compression method.

To set the compression level, the RMU SET command has been extended:

```
$ RMU /SET AFTER_JOURNAL /BACKUPS=(...,[ [NO]COMPRESSION[=ZLIB[=n]]])
  default is NOCOMPRESSION
  n = ZLIB compression level,
  default is 6, minimum is 1, maximum is 9
```

The RMU recover command has been enhanced to automatically decompress AIJ backup files in the 'Old_file' format which have been compressed using the ZLIB compression method.

13.1.11 Global Statistics Sections for Better Performance

On systems with many CPUs, updating database statistics from many application processes causes memory cache invalidation and therefore prolongs the update of the statistics data.

With the change in this release of Oracle Rdb, the RDM Monitor creates 16 global statistic sections for systems with 16 or more CPUs. Application processes attach to a statistics section based on the modulo 16 of their process ID value. This should reduce the coincidence that two or more processes

use the same global section from different processors and thus cause memory cache invalidation when updating statistics data.

The default used for RAD (Resource Allocation Domain) systems still remains (see below).

The use of multiple global statistic sections can be overridden with the following system logical name:

```
$ DEFINE /SYSTEM RDM$BIND_MONITOR_GLOBAL_STATS_SECTIONS n
n = 0 - always use statistics in the database's shared memory section
n = 1..16 - use statistics in separate global sections
           with n the number of global sections being used
If n is equal -1 or if the logical is not defined, use the default (see
below).
```

By default, the statistics area in the database's shared memory section is used unless a system has more than one RAD with memory or the system has 16 or more CPUs. In the case of more than one RAD with memory, one global statistics section is created per RAD with memory. In the case of 16 or more CPUs, 16 global statistics sections are created. The more than one RAD with memory case has precedence over the 16 or more CPUs case.

13.1.12 RMU/SET AUDIT Supports Wildcard Table and Column Names

Bug 5865199

In prior versions of Oracle Rdb, the RMU Set Audit command did not allow wildcards to be used to specify tables or column names for the DACCESS qualifier. A database administrator was required to enumerate all tables and columns to be audited. Further, if a table was specified as * then this was ignored by RMU.

With this release, Rdb RMU Set Audit has been enhanced to provide more flexible naming conventions for tables and columns.

- ◆ TABLE=* and COLUMN=*. * are now accepted to specify all tables or all columns of all tables.
- ◆ Table and column names may also contain OpenVMS style wildcards "*" and "%". For instance, in the PERSONNEL database, JOB* will match both the JOBS and JOB_HISTORY table and *HISTORY.EMPLOYEE_ID will match all EMPLOYEE_ID columns in any table ending in HISTORY which includes both the JOB_HISTORY and SALARY_HISTORY tables.

The following example shows the simplified commands for enabling auditing for important fields in the PERSONNEL database.

```
$ rmu/set audit-
  /enable=daccess=column=(-
    *.employee_id,-
    *_history.*end,-
    *_history.*start)-
  /privileges=(insert,select,update,delete) personnel
$
$ rmu/show audit/daccess=(DATABASE,TABLE,COLUMN) personnel
```

```

Security auditing STOPPED for:
  DACCESS (disabled)
    DATABASE
      (NONE)
    COLUMN : DEGREES.EMPLOYEE_ID
      (SELECT,INSERT,UPDATE,DELETE)
    COLUMN : EMPLOYEES.EMPLOYEE_ID
      (SELECT,INSERT,UPDATE,DELETE)
    COLUMN : JOB_HISTORY.EMPLOYEE_ID
      (SELECT,INSERT,UPDATE,DELETE)
    COLUMN : JOB_HISTORY.JOB_START
      (SELECT,INSERT,UPDATE,DELETE)
    COLUMN : JOB_HISTORY.JOB_END
      (SELECT,INSERT,UPDATE,DELETE)
    COLUMN : RESUMES.EMPLOYEE_ID
      (SELECT,INSERT,UPDATE,DELETE)
    COLUMN : SALARY_HISTORY.EMPLOYEE_ID
      (SELECT,INSERT,UPDATE,DELETE)
    COLUMN : SALARY_HISTORY.SALARY_START
      (SELECT,INSERT,UPDATE,DELETE)
    COLUMN : SALARY_HISTORY.SALARY_END
      (SELECT,INSERT,UPDATE,DELETE)

```

```

Security alarms STOPPED for:
  DACCESS (disabled)
    DATABASE
      (NONE)
    COLUMN : DEGREES.EMPLOYEE_ID
      (SELECT,INSERT,UPDATE,DELETE)
    COLUMN : EMPLOYEES.EMPLOYEE_ID
      (SELECT,INSERT,UPDATE,DELETE)
    COLUMN : JOB_HISTORY.EMPLOYEE_ID
      (SELECT,INSERT,UPDATE,DELETE)
    COLUMN : JOB_HISTORY.JOB_START
      (SELECT,INSERT,UPDATE,DELETE)
    COLUMN : JOB_HISTORY.JOB_END
      (SELECT,INSERT,UPDATE,DELETE)
    COLUMN : RESUMES.EMPLOYEE_ID
      (SELECT,INSERT,UPDATE,DELETE)
    COLUMN : SALARY_HISTORY.EMPLOYEE_ID
      (SELECT,INSERT,UPDATE,DELETE)
    COLUMN : SALARY_HISTORY.SALARY_START
      (SELECT,INSERT,UPDATE,DELETE)
    COLUMN : SALARY_HISTORY.SALARY_END
      (SELECT,INSERT,UPDATE,DELETE)

```

Usage Notes

When using wildcard characters with /ENABLE=DACCESS or /DISABLE=DACCESS option, only user defined objects will be selected.

- ◆ TABLE – only user defined tables will be matched by the wildcard search. Views, system tables and special tables created by OCI Services for Rdb will not be returned. For excluded tables, you must specify their full name on the RMU command line.
- ◆ SEQUENCE – only user defined sequences will be matched by the wildcard search. This includes sequences implicitly created for IDENTITY columns. For excluded sequences, you must specify their full name on the RMU command line.
- ◆ MODULE – only user defined modules will be matched by the wildcard search. For excluded modules, you must specify their full name on the RMU command line.

- ◆ ROUTINE – only user defined routines will be matched by the wildcard search. Any routine defined in a module with USAGE IS LOCAL will be excluded from this matching. Such routines can only be called from within the module itself and so additional auditing is not required. For excluded routines, you must specify their full name on the RMU command line.
- ◆ VIEW – only user defined views will be matched by the wildcard search. Base tables, system views, and special views created by OCI Services for Rdb will not be returned. For excluded views, you must specify their full name on the RMU command line.

13.1.13 RMU/BACKUP Database Root Verification Performance Enhancement

By default, RMU/BACKUP verifies the database root at the start of the backup before backing up an Oracle Rdb database. If the database root is invalid, the error diagnostics from the verification are output and the backup is terminated. This is to prevent backing up a corrupt database. To not verify the database root, the user must specify /NODATABASE_VERIFICATION. As part of this verification, all live and snapshot database storage areas were opened to verify the area prologue block and the area maximum page number and then closed. Later, the storage areas were again opened and closed a second time to do the actual backup of the data in each storage area.

To improve the performance of RMU/BACKUP, the extra open and close of the live and snapshot database storage areas just to verify the database root has been eliminated. The same verification is still done but the storage areas are now only opened and closed once during the backup. Note that /DATABASE_VERIFICATION remains the default for RMU/BACKUP.

As part of this change, the following new syntax has been added to the RMU/BACKUP/PARALLEL *.PLAN file.

```
[No]Database_Verification
```

The existing command line qualifier "/[NO]DATABASE_VERIFICATION" is used to set the new "[No]Database_Verification" option in the initial *.PLAN file created from the RMU/BACKUP command line. The *.PLAN file can then be edited to change this option if desired. The default is "/DATABASE_VERIFICATION".

The following example shows some of the verification diagnostics that can be output when the database root is verified by RMU/BACKUP.

```
$ RMU/BACKUP/NOLOG MF_PERSONNEL MFP.RBF
%RMU-E-BADMAXPNO, unable to read last page (1052) in file
DEVICE:[DIRECTORY]DEPARTMENTS.RDA;1
%RMU-F-ABORTVER, fatal error encountered; aborting verification
%RMU-F-FATALERR, fatal error on BACKUP
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 11-MAY-2009
11:16:59.46
$ RMU/BACKUP/NOQUIET/ONLINE/NOLOG MF_PERSONNEL MFP
%RMU-W-BADPROID, DEVICE:[DIRECTORY]EMPIDS_MID.SNP;1
file contains a bad identifier
Expected "RDMSDATA", found "NOTRDBDB"
%RMU-W-INVALFILE, inconsistent database file
DEVICE:[DIRECTORY]EMPIDS_MID.SNP;1
%RMU-F-ABORTVER, fatal error encountered; aborting verification
%RMU-F-FATALERR, fatal error on BACKUP
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 11-MAY-2009
```

11:11:25.68

The following example shows that the RMU/BACKUP *.PLAN file "[No]Database_Verification" option is set based on the "/DATABASE_VERIFICATION" qualifier on the RMU/BACKUP command line. The default is "Database_Verification" so unless "/NODATABASE_VERIFICATION" is specified, the "Database_Verification" option will be set in the *.PLAN file.

```
$ RMU/BACKUP/PARALLEL=EXEC=1/DISK=WRITER=2/LIST=TMP.PLAN/ONLINE-
MF_PERSONNEL DISK:[DIRECTORY1]MFP,DISK:[DIRECTORY2]
$ SEAR TMP.PLAN DATABASE_VERIFICATION
   Database_Verification
$ RMU/BACKUP/PARALLEL=EXEC=1/DISK=WRITER=2/LIST=TMP.PLAN/ONLINE-
/DATABASE_VERIFICATION MF_PERSONNEL DISK:[DIRECTORY1]MFP,-
DISK:[DIRECTORY2]
$ SEAR TMP.PLAN DATABASE_VERIFICATION
   Database_Verification
$ RMU/BACKUP/PARALLEL=EXEC=1/DISK=WRITER=2/LIST=TMP.PLAN/ONLINE-
/NODATABASE_VERIFICATION MF_PERSONNEL -
DISK:[DIRECTORY1]MFP,DISK:[DIRECTORY2]
$ SEAR TMP.PLAN DATABASE_VERIFICATION
   NoDatabase_Verification
$ RMU/BACKUP/PLAN TMP.PLAN
```

13.1.14 RMU /UNLOAD /AFTER_JOURNAL New Qualifier /DELETES_FIRST

Bug 3388774

The qualifier "/DELETES_FIRST" has been added to the RMU /UNLOAD /AFTER_JOURNAL command. Specifying "/DELETES_FIRST" indicates that all delete operations within each transaction are to be returned before any add/modify operations.

Record Order Unpredictable

Within the output stream for a transaction, the order of records returned from the LogMiner remains unpredictable.

13.1.15 Add Option to Pass Values to /CONFIRM During RESTORE Operation

Bug 411144

In prior releases of Oracle Rdb, if problems occurred during tape operation of an RMU/RESTORE command and the /CONFIRM option was selected, then the operation would wait for user input on the terminal before continuing.

```
$RMU/RESTORE/NOCCD/REWIND LMA1001:VOL002.RBF MF_PERSONNEL /DIRECTORY=[] /LOG
%RMU-I-WRNGLBL, Tape on LMA1001 was incorrectly labeled. Expected VOL002 -
Found MF_PER
```

Oracle® Rdb for OpenVMS

```
%RMU-I-TAPEDISPR, Specify tape disposition for LMA1001 (QUIT,OVERRIDE,RETRY,
UNLOAD)
RMU> QUIT (User has to enter the RESPONSE.)
%RMU-F-ABORT, operator requested abort on fatal error
%RMU-F-FATALERR, fatal error on RESTORE
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 13-JUL-2013 11:22:32.90

$RMU/RESTORE/NOCCD/REWIND LMA1001:VOL002.RBF MF_PERSONNEL /DIRECTORY=[] /LOG
%RMU-I-WRNGLBL, Tape on LMA1001 was incorrectly labeled. Expected VOL002 -
Found MF_PER
%RMU-I-TAPEDISPR, Specify tape disposition for LMA1001 (QUIT,OVERRIDE,RETRY,
UNLOAD)
RMU> RETRY (User has to enter the RESPONSE.)
%MOUNT-I-MOUNTED, MF_PER mounted on LMA1001:
%RMU-I-WRNGLBL, Tape on LMA1001 was incorrectly labeled. Expected VOL002 -
Found MF_PER
%RMU-I-TAPEDISPR, Specify tape disposition for LMA1001 (QUIT,OVERRIDE,RETRY,
UNLOAD)
RMU> QUIT (User has to enter the RESPONSE.)
%RMU-F-ABORT, operator requested abort on fatal error
%RMU-F-FATALERR, fatal error on RESTORE
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 13-JUL-2013 11:22:55.86

$RMU/RESTORE/NOCCD/REWIND LMA1001:VOL002.RBF MF_PERSONNEL /DIRECTORY=[] /LOG
%RMU-I-WRNGLBL, Tape on LMA1001 was incorrectly labeled. Expected VOL002 -
Found MF_PER
%RMU-I-TAPEDISPR, Specify tape disposition for LMA1001 (QUIT,OVERRIDE,RETRY,
UNLOAD)
RMU> OVERRIDE (User has to enter the RESPONSE.)
%RMU-F-FILACCERR, error opening input file LMA1001:[000000]VOL002.RBF;
-SYSTEM-W-NOSUCHFILE, no such file
%RMU-F-FATALERR, fatal error on RESTORE
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 13-JUL-2013 11:23:05.59
```

A new feature has been added in this release to correct this problem. The user has the option of selecting values for /CONFIRM during a RESTORE from tape operation. The new syntax and valid values are:

```
RMU/RESTORE... /CONFIRM[=QUIT|RETRY=x|OVERRIDE|UNLOAD]
```

See the following examples of this new feature.

```
$RMU/RESTORE/NOCCD/REWIND LMA1001:VOL002.RBF MF_PERSONNEL /DIRECTORY=[]
/LOG /CONFIRM=QUIT
%MOUNT-I-MOUNTED, MF_PER mounted on LMA1001:
%RMU-I-TAPEDEF, Terminating restore operation as requested by user
%RMU-F-ABORT, operator requested abort on fatal error
%RMU-F-FATALERR, fatal error on RESTORE
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 13-JUL-2013 11:43:31.35

$RMU/RESTORE/NOCCD/REWIND LMA1001:VOL002.RBF MF_PERSONNEL /DIRECTORY=[]
/LOG /CONFIRM=RETRY=2
%RMU-I-TAPEDEF, Retrying tape operation as requested by user
%MOUNT-I-MOUNTED, MF_PER mounted on LMA1001:
%RMU-I-TAPEDEF, Retrying tape operation as requested by user
%MOUNT-I-MOUNTED, MF_PER mounted on LMA1001:
%RMU-F-ABORT, operator requested abort on fatal error
%RMU-F-FATALERR, fatal error on RESTORE
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 13-JUL-2013 11:43:42.97
```

```

$RMU/RESTORE/NOCCD/REWIND LMA1001:VOL002.RBF MF_PERSONNEL /DIRECTORY=[]
/LOG /CONFIRM=OVERRIDE
%RMU-I-TAPEDEF, Overriding tape label as requested by user
%RMU-F-FILACCERR, error opening input file LMA1001:[000000]VOL002.RBF;
-SYSTEM-W-NOSUCHFILE, no such file
%RMU-F-FATALERR, fatal error on RESTORE
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 13-JUL-2013 11:43:58.38

```

13.1.16 Table Names Can Now Be Specified For Index Verification

A new feature has been added to the Oracle Rdb RMU/VERIFY command which allows table names to be specified for database index verification. Currently, only a list of one or more index names can be specified with either the /INDEXES or /INDICES qualifier. Now, if one or more database table names is specified with the new /FOR_TABLE qualifier, all the indexes defined for the named table(s) will be selected for verification. If the /FOR_TABLE qualifier is used, then either the /INDEXES or /INDICES qualifier must also be specified in the same RMU/VERIFY command.

The /INDEXES or /INDICES qualifiers can continue to specify a list of one or more index names to be verified in addition to the /FOR_TABLE list of one or more table names for which all the indexes defined for each table are to be verified. Index names cannot be specified with the new /FOR_TABLE qualifier and table names cannot be specified with the existing /INDEXES or /INDICES qualifiers. Either the syntax RMU/VERIFY/INDEXES/FOR_TABLE=* or RMU/VERIFY/INDEXES/FOR_TABLE can be used to specify that all indexes defined for all database tables should be verified. If lower case characters are not to be converted to upper case in table names, table names must be delimited with double quotes.

The following new syntax can be specified for the /FOR_TABLE qualifier.

```

/FOR_TABLE=(table_name,...)
/FOR_TABLE=table_name
/FOR_TABLE=*
/FOR_TABLE

```

The following examples show that both the /FOR_TABLE and /INDEXES or /INDICES qualifiers can specify either no values or lists of one or more table or index names in the same RMU/VERIFY command line. Only table name values can be specified with the /FOR_TABLE qualifier, and only index name values can be specified with the /INDEXES or /INDICES qualifiers. If table name values are specified, all the indexes defined for each named table will be verified.

```

$RMU/VERIFY/INDEXES/FOR_TABLE=EMPLOYEES/NOLOG MF_PERSONNEL
$RMU/VERIFY/INDEXES=(EMPLOYEES_HASH,EMP_EMPLOYEE_ID)/FOR_TABLE=COLLEGES/NOLOG
MF_PERSONNEL
$RMU/VERIFY/INDICES=EMP_EMPLOYEE_ID/FOR_TABLE=(COLLEGES,JOB_HISTORY)/NOLOG
MF_PERSONNEL

```

The following examples show that both of the following commands are equivalent and will verify all indexes defined for all tables in the database.

```

$RMU/VERIFY/INDEXES/FOR_TABLE=*/NOLOG MF_PERSONNEL
$RMU/VERIFY/INDICES/FOR_TABLE/NOLOG MF_PERSONNEL

```

The following examples shows that if the /FOR_TABLE qualifier is specified, either the /INDEXES or /INDICES qualifiers must be specified in the same RMU/VERIFY command or a fatal error will occur.

```
$RMU/VERIFY/FOR_TABLE=EMPLOYEES MF_PERSONNEL
%RMU-F-CONFLSWIT, conflicting qualifiers /FOR_TABLE and /INDEXES or /INDICES
not specified
%RMU-F-FTL_VER, Fatal error for VERIFY operation at 14-FEB-2013 14:58:47.09
```

13.1.17 New RMU/VERIFY Feature to Detect Orphan Hash Index Buckets

To ensure Oracle Rdb database integrity, a new feature has been added to the RMU/VERIFY command to detect "orphan" hash index buckets on database pages in mixed storage areas which do not belong to any existing hash index defined for the database. Orphan hash buckets are either not referenced in a SYSTEM RECORD on a mixed area database page or are not referenced by another hash bucket.

This is not a default feature but must be activated by specifying the new "/ORPHAN_INDEXES" qualifier on the command line. Orphan hash index buckets will only be reported if RMU/VERIFY is verifying one or more hash indexes as part of the database verification. For each orphan hash bucket detected, an error message will be output specifying the storage area name and physical "dbkey" address of the orphan hash bucket. The physical dbkey address specifies the storage area number, the storage area page number, and the storage area line number of the orphan hash bucket. This information can be used with the RMU/DUMP command to dump the area page where the orphan hash bucket is located.

In the short term, these orphaned hash index elements are harmless but consume space which would otherwise be used by new inserts. Eventually, as objects get dropped and created, these elements may be confused with current structures. Therefore, Oracle recommends cleaning them up as soon as practical.

However, these orphaned hash index elements can no longer be removed using the standard DROP commands in SQL. To reclaim the space used by these elements will require a DROP STORAGE AREA for the affected area. The database administrator should create a replacement storage area and use ALTER or DROP commands to move other tables and indices out of the affected storage area. Then use DROP STORAGE AREA to remove the unused area. Alternatively, you can use the SQL EXPORT DATABASE and IMPORT DATABASE commands to rebuild the whole database.

If RMU/VERIFY detects and reports any structural problems with the database pages or hash index structures for the area being verified, or any problems with VMS sort which is used in the process of detecting orphan hash buckets, detection of orphan hash buckets will be aborted for the area and the following error message will be output.

```
$ RMU/VERIFY/ALL/ORPHAN_INDEXES MF_PERSONNEL
%RMU-E-ORPHANERR, Error searching for orphan index nodes in area EMPIDS_LOW
```

The user should correct the reported problems and repeat the verification.

The new syntax for this feature which can be specified with the RMU/VERIFY command is the following.

```
/[NO]ORPHAN_INDEXES
```

The default is "/NOORPHAN_INDEXES" – orphan hash buckets will not be detected or reported. To activate this feature "/ORPHAN_INDEXES" must be specified.

The following example shows the diagnostic error message that will be put out by RMU/VERIFY for each orphan hash bucket found on a database page in the current storage area. Both the storage area name and the physical dbkey address of the orphan hash bucket are reported. The dbkey address in the message following the word "at" specifies the storage area number followed by the page number followed by the line number where the orphan hash bucket is located.

```
$ RMU/VERIFY/ALL/ORPHAN_INDEXES/NOERROR_LIMIT DATABASE.RDB
%RMU-E-ORPHANIDX, Orphan hashed index bucket found in area
  DATABASE_AREA at 21:2:5
%RMU-E-ORPHANIDX, Orphan hashed index bucket found in area
  DATABASE_AREA at 21:2:9
%RMU-E-ORPHANIDX, Orphan hashed index bucket found in area
  DATABASE_AREA at 21:2510:32
%RMU-E-ORPHANIDX, Orphan hashed index bucket found in area
  DATABASE_AREA at 21:2510:43
```

13.1.18 New COMPILE Clause for ALTER TRIGGER Statement

This release of Oracle Rdb supports a new COMPILE clause for the ALTER TRIGGER statement. This clause directs Rdb to re-compile the trigger to ensure that it is valid. If COMPILE is successful and the trigger was marked invalid, but "Can be revalidated" then the invalid flag will be cleared.

Triggers can be marked "invalid" when a procedure, function or sequence is dropped using the CASCADE clause.

The following example shows how this new clause could be used.

```
SQL> set flags 'warn_invalid';
SQL>
SQL> alter module M
cont>      drop procedure PP cascade;
~Xw: Trigger "C_INSERT" marked invalid
SQL>
SQL> show trigger C_INSERT
      C_INSERT
Current state is INVALID
      Can be revalidated
Source:
c_insert
  after insert on C
  when (C.b is NULL)
    (call PP())
  for each row
SQL>
SQL> alter trigger C_INSERT
```

```

cont>      compile;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-BAD_SYM, unknown routine symbol - PP
SQL>
SQL> alter module M
cont>      add procedure PP;
cont>      trace 'in PP';
cont> end module;
SQL>
SQL> alter trigger C_INSERT
cont>      compile;
SQL>
SQL> show trigger C_INSERT
      C_INSERT
Source:
c_insert
  after insert on C
  when (C.b is NULL)
    (call PP())
  for each row
SQL>

```

Note

*Any trigger marked with the invalid flag will still be used by Oracle Rdb if at runtime it can be compiled successfully. However, only the **COMPILE** clause of the **ALTER TRIGGER** statement, or the **COMPILE ALL TRIGGERS** clause of the **ALTER TABLE** statement will clear the "invalid" flag.*

13.1.19 New COMPILE ALL TRIGGERS Clause for ALTER TABLE Statement

This release of Oracle Rdb supports a new COMPILE ALL TRIGGERS clause for the ALTER TABLE statement. This clause directs Rdb to re-compile all the triggers defined for the table to ensure that they are valid. If COMPILE ALL TRIGGERS is successful and any trigger was marked invalid and "Can be revalidated" then the invalid flag will be cleared.

Triggers can be marked "invalid" when a procedure, function or sequence is dropped using the CASCADE clause.

The following example shows how this new clause could be used.

```

SQL> set flags 'warn_invalid';
SQL>
SQL> alter module M
cont>      drop procedure PP cascade;
~Xw: Trigger "C_INSERT" marked invalid
SQL>
SQL> show trigger C_INSERT
      C_INSERT
Current state is INVALID
  Can be revalidated
Source:
c_insert

```

```

    after insert on C
    when (C.b is NULL)
      (call PP())
    for each row
SQL>
SQL> alter table C
cont>      compile all triggers;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-BAD_SYM, unknown routine symbol - PP
SQL>
SQL> ! Replace missing procedure PP
SQL> alter module M
cont>      add procedure PP;
cont>      trace 'in PP';
cont> end module;
SQL>
SQL> alter table C
cont>      compile all triggers;
SQL>
SQL> ! Show that the INVALID flag is now cleared
SQL> show trigger C_INSERT
      C_INSERT
Source:
c_insert
  after insert on C
  when (C.b is NULL)
    (call PP())
  for each row
SQL>

```

Note

*Any trigger marked with the invalid flag will still be used by Oracle Rdb if at runtime it can be compiled successfully. However, only the **COMPILE** clause of the **ALTER TRIGGER** statement or the **COMPILE ALL TRIGGERS** clause of the **ALTER TABLE** statement will clear the "invalid" flag.*

13.1.20 New RETRY Clause for ACCEPT Statement

This release of Oracle Rdb adds a new RETRY clause to the ACCEPT Statement. The RETRY clause specifies the number of times that SQL will reprompt the user when an error occurs after processing the user's input.

The following example shows that after an erroneous input, the user is prompted again for the data.

```

SQL> declare :v integer = 0;
SQL> accept :v retry 5;
Enter value for V: xxxx
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-COSI-F-INPCONERR, input conversion error
Enter value for V: 42
SQL> print :v;
      V
      42
SQL>

```

13.1.21 New Character Sets ISOLATIN2 and WIN_LATIN2 Supported

This release of Oracle Rdb adds two new character sets, ISOLATIN2 and WIN_LATIN2.

Usage Notes

ISOLATIN2 is a single octet character set that has the following characteristics:

- ◆ Encodes Extended East European characters as defined by the ISO/IEC 8859–2 standard.
- ◆ Fixed single octet characters.
- ◆ May be used as an Identifier character set.
- ◆ Contains the full set of ASCII characters.
- ◆ EE8ISO8859P2 is the Oracle NLS equivalent character set.
- ◆ The translation name to translate to ISOLATIN2 characters is RDB\$ISOLATIN2.
- ◆ The Wildcard Underscore character is %X5F.
- ◆ The Wildcard Percent character is %X25.

WIN_LATIN2 is a single octet character set that has the following characteristics:

- ◆ Encodes Extended East European characters as defined by the MS Windows Code Page 1250 8–Bit standard.
- ◆ Fixed single octet characters.
- ◆ May be used as an Identifier character set.
- ◆ Contains the full set of ASCII characters.
- ◆ EE8MSWIN1250 is the Oracle NLS equivalent character set.
- ◆ The translation name to translate to WIN_LATIN2 characters is RDB\$WIN_LATIN2.
- ◆ The Wildcard Underscore character is %X5F.
- ◆ The Wildcard Percent character is %X25.

13.1.22 Changes and Enhancements to Trigger Support

In this release of Oracle Rdb, the handling of trigger definitions has been changed to allow future enhancements to triggers.

The following changes may be observed with this release.

- ◆ In prior versions, the entire definition was stored in a single row in the system table Rdb\$TRIGGERS. With this release, all new trigger definitions are split into separate rows stored in Rdb\$TRIGGER_ACTIONS with a single base row in Rdb\$TRIGGERS.
- ◆ Each trigger action is given a generated action name.
- ◆ Each trigger is assigned a unique trigger identification.
- ◆ Trigger definitions that existed in prior releases of Oracle Rdb will remain unchanged in a database converted to Oracle Rdb Release 7.3 using RMU/CONVERT or RMU/RESTORE (which implicitly calls RMU/CONVERT). All new triggers created with Oracle Rdb Release 7.3 or later will use the new format.
- ◆ Oracle Rdb no longer supports the export of Triggers from remote databases older than Rdb V6.0. These would be older systems running on VAX and Alpha systems with Rdb V5.1 or earlier. The EXPORT DATABASE will need to be run under that Rdb release and not remotely from an Oracle Rdb V7.3 system.

It should be noted that the SQL (and RDO) EXPORT DATABASE statement will now save extended attributes. Therefore, interchange files created with Oracle Rdb V7.3 used with older versions will not support new trigger definitions. Oracle Rdb V7.2.5.3 or later is required to handle the new interchange format.

The following example shows the errors reported when triggers created using Rdb V7.3 and exported, are imported using an older Rdb V7.2 release.

```
SQL> IMPORT DATABASE FROM TP_EXP.RBR FILENAME TP2;
%SQL-F-NOTRGRES, Unable to IMPORT trigger TELLER_DELETE
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-BAD_CODE, corruption in the query string
%SQL-F-NOTRGRES, Unable to IMPORT trigger TELLER_INSERT
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-BAD_CODE, corruption in the query string
SQL>
```

Oracle recommends using RMU/EXTRACT/ITEM=TRIGGERS if the triggers need to be recreated in prior versions. RMU Extract is a best-effort utility and some manual editing of the generated SQL syntax may be required.

13.1.23 New RMU BACKUP RBF File BRH\$K_ROOT1, BRH\$K_ROOT2, BRH\$K_ROOT3 Records /kroot_records

In this release of Oracle Rdb, three new BRH record types have been added to the RBF file used to back up Oracle Rdb databases. New BRH\$K_ROOT1, BRH\$K_ROOT2, and BRH\$K_ROOT3 records have been added for backing up the enlarged Rdb root file KROOT structure in three parts. This will preserve the current minimum /BLOCK_SIZE of 2048 that can be specified for determining the buffer size used for backing up BRH records to the backup RBF file. For this release of Rdb, the KROOT has been enlarged to 5120 bytes which will not fit into the smaller block sizes that can be specified with the optional /BLOCK_SIZE qualifier for the RMU /BACKUP, DUMP/BACKUP, /BACKUP/AFTER_JOURNAL and /OPTIMIZE/AFTER_JOURNAL commands. Since the enlarged KROOT is now backed up and restored in three parts, the current range of between 2048 and 65,024 bytes that can be specified with the optional /BLOCK_SIZE qualifier has not changed for this release.

The BRH record type previously used to back up the smaller 1536 byte Rdb KROOT root structure for Rdb V7.2 and earlier releases in one record was BRH\$K_ROOT. This record type will no longer be in RBF backup files produced by RMU BACKUP commands created by this version of Rdb, but it will be accepted by the RMU/RESTORE and RMU/DUMP/BACKUP commands which currently accept RBF files produced by previous V72, V71 and V70 versions of Oracle Rdb. However, V72 and previous versions of Oracle Rdb will not accept RBF records produced by this release but will return the error *%RMU-E-INVRECTYP, invalid record type in backup file* for the new BRH\$K_ROOT1, BRH\$K_ROOT2 and BRH\$K_ROOT3 backup record types.

The following example shows the Oracle Rdb V7.3 database backup file MFP73.DMP created by the RMU/BACKUP command, which is then dumped with the most detailed "DEBUG" option by the RMU/DUMP/BACKUP command. The portion of the dump file shown contains the new BRH\$K_ROOT1 (TYPE = 32), BRH\$K_ROOT2 (TYPE = 33) and BRH\$K_ROOT3 (TYPE = 34) records now used to backup the enlarged Rdb root file KROOT structure.

Oracle® Rdb for OpenVMS

\$ RMU/BACKUP MF_PERSONNEL.RDB MFP73.RBF
\$ RMU/DUMP/BACKUP/OPTION=DEBUG/OUTPUT=MFP73.DMP MFP73.RBF
\$ TYPE MFP73.DMP

REC_SIZE = 1708 REC_TYPE = 32 BADDATA = 00
ROOT = 01 AREA_ID = 0 LAREA_ID = 0
PNO = 0

000000000000000000000000A002006AC 0000 '.

KODA database root record

000000000000049544F4F52534D4452 0000 'RDMSROOTI.....'
2EC700C900A66EB9EBCF255B00000000 0010 '.....[%İë¹n.É.Ç.'
0000002100000088000000170000000F 0020 '.....!....'
0000000B000000280000001900000022 0030 '".....(.....'
00000070000000800000003A0000006C 0040 '1...:.....p...'

00000010000000040000001200000001 0050 '.....'
00000014000000140000001000000032 0060 '2.....'
0000000300000005000000FA00000006 0070 '.....ú.....'
000000000000000A0000000A0000000A 0080 '.....'
00000000000000000000000000000000 0090 '.....'
00000100000000000000000000000000 00A0 '.....'
00000000000000000000000000000000 00B0 '.....'

::: (1 duplicate line)

000000300000002000000020000000 00D0 '.....'
000000080000000800AC93EB3CA7391C 00E0 '.9§<ë.....'
00000001000000040000000500000005 00F0 '.....'
00000000000000040000002400000002 0100 '.....\$......'
00000004000000010000008A00000000 0110 '.....'
0000000000000000000000FF00000004 0120 '.....'
00000200000000000000000000000000 0130 '.....'
20571AEA00000000000000200000008B 0140 '.....ê.W'

0000000000AC93EB216424E900AC93EB 0150 'ë...é\$ð!ë.....'
00000000000000000000000000000000 0160 '.....'
0000008C000000000000000000000000 0170 '.....'
0000000000000000000000000140000010 0180 '.....'
00000000000000000000000000000000 0190 '.....'
00AC93EB205B2CA40000000000000000 01A0 '.....,[ë...'

00000000000000000000000000000000 01B0 '.....'

::: (20 duplicate lines)

4F485B3A31524553555F424452455347 0300 'GSERDB_USER1:[HO'
37562E545245564E4F432E494C554843 0310 'CHULI.CONVERT.V7'
545345542E544944452E4B524F572E33 0320 '3.WORK.EDIT.TEST'
4E4E4F535245505F464D5D504C45482E 0330 '.HELP]MF_PERSONN'
0000000000000000313B4244522E4C45 0340 'EL.RDB;1.....'
00000000000000000000000000000000 0350 '.....'

::: (52 duplicate lines)

00000000000000000000000000000000 06A0 '.....'

REC_SIZE = 1706 REC_TYPE = 33 BADDATA = 00
ROOT = 01 AREA_ID = 0 LAREA_ID = 0
PNO = 0

000000000000000000000000A002106AA 0000 'a.!.....'

KODA database root record

00000000000000000000000000000000 0000 '.....'

::: (105 duplicate lines)

00000000000000000000000000000000 06A0 '.....'

REC_SIZE = 1706 REC_TYPE = 34 BADDATA = 00
ROOT = 01 AREA_ID = 0 LAREA_ID = 0

```
PNO = 0
```

```
00000000000000000000000000000000A002206AA 0000 'a.".....'
KODA database root record
000000000000000000000000000000000000 0000 '.....'
::: (105 duplicate lines)
00000000000000000000000000000000 06A0 '.....'
```

13.1.24 New Functions NUMTODSINTERVAL, NUMTOYMINTERVAL Supported

This release of Oracle Rdb adds two new functions for compatibility with the Oracle database.

◆ NUMTODSINTERVAL

NUMTODSINTERVAL (n, 'interval_unit')

NUMTODSINTERVAL converts n to an INTERVAL DAY TO SECOND value.

The first argument, n, can be any numeric value. The value for interval_unit specifies the interval qualifier and must resolve to one of the following string values: 'day', 'hour', 'minute', 'second'. Interval_unit is case insensitive. Leading and trailing values within the parentheses are ignored. The interval leading precision of the return is 9.

◆ NUMTOYMINTERVAL

NUMTOYMINTERVAL (n, 'interval_unit')

NUMTOYMINTERVAL converts n to an INTERVAL YEAR TO MONTH literal.

The first argument, n, can be any numeric value. The value for interval_unit specifies the interval qualifier and must resolve to one of the following string values: 'year', 'month'. Interval_unit is case insensitive. Leading and trailing values within the parentheses are ignored. The interval leading precision of the return is 9.

Usage Notes

- ◆ This function is implicitly converted by SQL to the equivalent CAST function. Therefore, other facilities such as RMU Extract or SET FLAGS with the STRATEGY,DETAIL options will show CAST only.

```
SQL> select last_name
cont> from employees
cont> where birthday + NUMTOYMINTERVAL (20, 'year') > current_date;
Tables:
  0 = EMPLOYEES
Conjunct: (0.BIRTHDAY + CAST (20 AS INTERVAL YEAR(9))) > CURRENT_DATE
Get      Retrieval sequentially of relation 0:EMPLOYEES
0 rows selected
```

Examples

This example queries the PERSONNEL database and lists any employees who are more than 20 years older than their manager.

```
SQL> select e.last_name || e.first_name as employee,
cont>         e.birthday,
cont>         m.last_name || m.first_name as manager,
cont>         m.birthday
cont> from job_history jh, employees e, departments d, employees m
cont> where jh.employee_id = e.employee_id
```

```

cont> and jh.job_end is null
cont> and jh.department_code = d.department_code
cont> and d.manager_id <> e.employee_id
cont> and d.manager_id = m.employee_id
cont> and e.birthday + NUMTOYMINTERVAL (20, 'year') < m.birthday;
EMPLOYEE          E.BIRTHDAY      MANAGER          M.BIRTHDAY
Iacobone          Eloi             1-May-1933      Stornelli        James            10-Jan-1960
Nash              Walter           19-Jan-1925     Keisling          Edward           21-Mar-1957
Hall              Lawrence         10-Jul-1933     Belliveau         Paul             9-May-1955
Clairmont         Rick             23-Dec-1924     Clarke            Karen            16-May-1950
Johnson          Bill             13-Apr-1927     Clarke            Karen            16-May-1950
5 rows selected
SQL>

```

13.1.25 RMU Dump Audit Command

When RMU/SET AUDIT is used to enable auditing for a database, Oracle Rdb writes records to the OpenVMS audit journal (for example SYS\$MANAGER:SECURITY.AUDIT\$JOURNAL). This command can be used to dump selected records from an OpenVMS AUDIT journal for a specific database for review.

This command is closely related to the RMU/LOAD/AUDIT command in that it reads and processes the rows from the audit journal.

Format

```
RMU/DUMP/AUDIT root-file-spec input-file-name
```

Command Qualifiers	Defaults
/BEFORE=timestamp	none
/FORMAT=formatting-options	/FORMAT=LIST
/LOG	/NOLOG
/OUTPUT=outputfile	/OUTPUT=SYS\$OUTPUT
/SINCE=timestamp	none
/TYPE=(type-list)	/TYPE=ALL

Command Parameters

- ◆ root-file-spec
The file specification for the database root file into which the table will be loaded. The default file extension is .rdb.
- ◆ input-file-name
The input-file-name parameter is the name of the journal containing the audit record data to be dumped. The default file extension is .AUDIT\$JOURNAL. You can determine the name of the security audit journal by using the DCL SHOW AUDIT/JOURNAL command.

Command Qualifiers

- ◆ Before=date-time
Specifies the ending date and time for records extracted from the audit journal. The value is a standard OpenVMS date and time. Enclose the date in quotes if it also includes a space between the date and time fields. If omitted, then all records to the end of the journal will be

dumped.

◆ **Format=formatting-options**

This qualifier allows the database administrator to change the default format (LIST) to XML. The XML output is more useful for archiving and historical analysis.

The following formatting options are accepted:

- ◇ LIST – the default output displays the attribute and value on a single line. Long values will be split across multiple lines if necessary.
- ◇ XML – formats the audit details as an XML record that can be archived.
- ◇ CHARACTER_ENCODING_XML – adjusts the character encoding to that appropriate to the data being dumped in XML format.
CHARACTER_ENCODING_XML is not compatible with the LIST keyword.

◆ **Log**

If specified, RMU will display a summary line reporting the number of records read from the audit journal and the count of those displayed by RMU Dump.

◆ **Since=date-time**

Specifies the starting date and time for records extracted from the audit journal. The value is a standard OpenVMS date and time. Enclose the date in quotes if it also includes a space between the date and time fields. If omitted, then all records from the start of the journal will be dumped.

◆ **Type=type-list**

Select different types of audit records. Values include: ALL, NONE, AUDIT, DACCESS, PROTECTION, and RMU. The list may contain negated values, such as /TYPE=(ALL,NORMU) so that some categories are removed. The default is to display all types of audit records.

◆ **Output[=files-spec]**

Specifies the name of the file where output is sent. The default is SYS\$OUTPUT. The default output file type is .LIS, if you specify a file name.

Usage Notes

- ◆ To use the RMU Dump Audit command you must have the RMU\$SECURITY privilege in the root file ACL for the database whose security audit records are being loaded. If you do not have this privilege, you must have the OpenVMS SYSPRV or BYPASS privilege.
- ◆ The OpenVMS audit journal may contain data from multiple facilities in addition to Oracle Rdb and may also contain audit records for other databases. Therefore, only a subset may be read and formatted by RMU Dump.
- ◆ Each audit record is divided into packets. Each packet contains a piece of audit information. The output from RMU Dump Audit displays the type for the packet (for instance NSA\$C_PKT_FINAL_STATUS), and the formatted value. If necessary, long text packets will be wrapped across multiple lines.
- ◆ Oracle Rdb uses a combination of OpenVMS types (those starting with NSA\$C) and Oracle Rdb tags (those starting with RDBNSA\$K). These tags are described below. Please refer to the relevant OpenVMS documentation for descriptions of the NSA\$C types).
- ◆ The width of the terminal session is used to limit the lines for the output to SYS\$OUTPUT.

Table 13-1 RDBNSA\$K types

Type	Description
RDBNSA\$K_PKT_DBNAME	File specification for the database root file
RDBNSA\$K_PKT_TSN	TSN (transaction sequence number) for the user process
RDBNSA\$K_PKT_DACCESS	

Oracle® Rdb for OpenVMS

	Discretionary access privileges for the user; based on object ACL, and OpenVMS override privileges
RDBNSA\$K_PKT_NEW_ACE	Result of a GRANT or REVOKE statement
RDBNSA\$K_PKT_OBJ_TYPE	Type of object being altered
RDBNSA\$K_PKT_OLD_ACE	Prior value before a GRANT or REVOKE statement
RDBNSA\$K_PKT_OPERATION_CODE	Description of the operation
RDBNSA\$K_PKT_RDB_PRIV_USED	Oracle Rdb privilege used for the operation
RDBNSA\$K_PKT_RMU_ARGS	The RMU command line for the operation
RDBNSA\$K_PKT_STATUS_CODE	OpenVMS condition following the operation attempt

Examples

Example 1: Dumping output of DACCESS records

The following example extracts just the DACCESS records since a known time.

```

$ RMU/DUMP/AUDIT-
  MF_PERSONNEL-
  SYS$MANAGER:SECURITY.AUDIT$JOURNAL-
  /TYPE=(NONE,DACCESS)-
  /LOG-
  /SINCE="14-MAR-2013 14:36:16.70"
.
.
.
-----:
REC_SUBTYPE                : DACCESS
RDBNSA$K_PKT_DBNAME        : _$1$DGA174:[SMITH.WORK.DB]MF_PE
                           : RSONNEL.RDB;1
NSA$C_PKT_AUDIT_NAME       : SECURITY
NSA$C_PKT_SYSTEM_ID        : 44262
NSA$C_PKT_IMAGE_NAME       : $1$DGA2:[SYS1.SYSCOMMON.][SYSEX
                           : E]SQL$73.EXE
NSA$C_PKT_PROCESS_ID       : 2B60A272
NSA$C_PKT_PROCESS_NAME     : Ian Smith
NSA$C_PKT_SYSTEM_NAME      : MALIBU
NSA$C_PKT_TIME_STAMP       : 14-MAR-2013 14:36:16.7079869
NSA$C_PKT_TERMINAL         : TNA38:
RDBNSA$K_PKT_TSN           : 0:9985
NSA$C_PKT_SUBJECT_OWNER    : [PROD,SMITH]
NSA$C_PKT_USERNAME         : SMITH
NSA$C_PKT_MESSAGE_TYPE_STR : Attempted table access
NSA$C_PKT_OBJECT_NAME      : JOB_HISTORY
RDBNSA$K_PKT_OBJ_TYPE      : TABLE
RDBNSA$K_PKT_OPERATION_CODE : Protection Change
RDBNSA$K_PKT_DACCESS       : SELECT,INSERT,UPDATE,DELETE,CRE
                           : ATE,ALTER,DROP,OPERATOR,DBADM,R
                           : EFERENCES
RDBNSA$K_PKT_PRIV_DESIRE    : DBCTRL
RDBNSA$K_PKT_STATUS_CODE   : Oracle Rdb privilege override
NSA$C_PKT_FINAL_STATUS     : %SYSTEM-S-NORMAL
RDBNSA$K_PKT_RDB_PRIV_USED : DBADM
-----:
%RMU-I-DATRECREAD, 6454 data records read from input file.
%RMU-I-DATRECUNL, 4 data records unloaded.

```

Oracle® Rdb for OpenVMS

Example 2: Dumping Output in XML Format

The following example extracts details for a time range in XML format.

```
$ RMU/DUMP/AUDIT -
  EVENTS_DB -
  SYS$MANAGER:SECURITY.AUDIT$JOURNAL -
  /FORMAT=XML -
  /SINCE="12-MAR-2013 16:02:01.40" -
  /BEFORE="12-MAR-2013 16:02:01.97" -
  /LOG

<?xml version="1.0" encoding="iso-8859-1"?>
<!-- RMU Dump Audit for Oracle Rdb V7.3-10 -->
<!-- Generated: 12-MAR-2013 16:10:16.15 -->
<!-- Database:  _$1$DGA174:[PRODUCTION.DATABASES]EVENTS_DB.RDB;1 -->
<!-- Since:  2013-03-12T16:02:01.4000000 -->
<!-- Before: 2013-03-12T16:02:01.9700000 -->

<audit>
<audit_record type="AUDIT">
<database_name>_1$DGA174:[PRODUCTION.DATABASES]EVENTS_DB.RDB;1</database_name>
<audit_name>SECURITY</audit_name>
  <system_id>44390</system_id>
  <image_name>DISK$VMSSYS:<SYS0.SYSCOMMON.SYSEXE>RMU73.EXE</image_name>
  <process_id>15928887</process_id>
  <process_name>DB_Admin</process_name>
  <system_name>PROD03</system_name>
  <time_stamp>2013-03-12T16:02:01.5802476</time_stamp>
  <terminal>TNA465:</terminal>
  <tsn>0</tsn>
  <subject_owner>[DBA,SMITH]</subject_owner>
  <username>SMITH      </username>
  <message_type>Auditing change</message_type>
  <rmu_command>RMU/SET AUDIT/TYPE=ALARM/START EVENTS_DB</rmu_command>
  <privilege_desired>RMU$SECURITY</privilege_desired>
  <status_code>RMU required privilege</status_code>
  <final_status>%SYSTEM-S-NORMAL</final_status>
  <rdb_privilege_used>RMU$SECURITY</rdb_privilege_used>
</audit_record>
</audit>
<!-- 1 row unloaded -->
%RMU-I-DATRECREAD, 207 data records read from input file.
%RMU-I-DATRECUNL,  1 data records unloaded 12-MAR-2013 16:10:16.16.
$ set noverify
```

Example 3: Dumping output in LIST format

The following example extracts details for a time range in LIST format.

```
$ RMU/DUMP/AUDIT -
  EVENTS_DB -
  SYS$MANAGER:SECURITY.AUDIT$JOURNAL -
  /TYPE=(AUDIT) -
  /SINCE="12-MAR-2013 16:02:01.40" -
  /BEFORE="12-MAR-2013 16:02:01.97" -
  /LOG

REC_SUBTYPE           : AUDIT
RDBNSA$K_PKT_DBNAME   :  _1$DGA174:[PRODUCTION.DATABASES]EVENTS_DB.RDB;1
NSA$C_PKT_AUDIT_NAME  : SECURITY
```

Oracle® Rdb for OpenVMS

```
NSA$C_PKT_SYSTEM_ID          : 44390
NSA$C_PKT_IMAGE_NAME        : DISK$VMSSYS:<SYS0.SYSCOMMON.SYSEXE>RMU73.EXE
NSA$C_PKT_PROCESS_ID       : 215ECCE5
NSA$C_PKT_PROCESS_NAME     : DB_Admin
NSA$C_PKT_SYSTEM_NAME      : PROD03
NSA$C_PKT_TIME_STAMP       : 2013-03-12 16:02:01.5802476
NSA$C_PKT_TERMINAL        : TNA465:
RDBNSA$K_PKT_TSN           : 0:0
NSA$C_PKT_SUBJECT_OWNER    : [DBA,SMITH]
NSA$C_PKT_USERNAME         : SMITH
NSA$C_PKT_MESSAGE_TYPE_STR : Auditing change
RDBNSA$K_PKT_RMU_ARGS      : RMU/SET AUDIT/TYPE=ALARM/START EVENTS_DB
RDBNSA$K_PKT_PRIV_DESIRED  : RMU$SECURITY
RDBNSA$K_PKT_STATUS_CODE   : RMU required privilege
NSA$C_PKT_FINAL_STATUS     : %SYSTEM-S-NORMAL
RDBNSA$K_PKT_RDB_PRIV_USED : RMU$SECURITY
-----:
%RMU-I-DATRECREAD, 207 data records read from input file.
%RMU-I-DATRECUNL, 1 data records unloaded 12-MAR-2013 16:19:22.31.
$
```

13.1.26 New BIN_TO_NUM Numeric Function

The function BIN_TO_NUM converts a bit vector to its equivalent number. Each argument to this function represents a bit in the bit vector (the last argument is the least significant bit of the number). This function takes as arguments any numeric datatype. Each expression must evaluate to 0 or 1. This function returns a BIGINT value with zero scale. If any argument evaluates to NULL, then the result of the conversion is NULL. This function accepts from 1 to 64 arguments.

Syntax

```
--> BIN_TO_NUM ( --> value_expr --> ) ---->
                |               |
                +-----, <-----+
```

Examples

The following example shows the result from using BIN_TO_NUM.

```
SQL> select bin_to_num (x, y, z), x, y, z from bin_tab order by 1;

          0          X          Y          Z
          1          0          0          1
          2          0          1          0
          3          0          1          1
          4          1          0          0
          5          1          0          1
          6          1          1          0
          7          1          1          1

8 rows selected
SQL>
```

13.1.27 RMU /PROGRESS_REPORT and Control-T for RMU Backup and Restore

This release of Oracle Rdb adds a new feature to display the performance and progress of backup and restore operations. This feature can be activated by typing Control-T after the RMU Backup or Restore operation has been started from the command line or by adding /PROGRESS_REPORT to the RMU command line.

The use of Control-T has to be enabled at the DCL level using:

```
$ SET CONTROL=T
```

While a Control-T just displays the information once, the /PROGRESS_REPORT qualifier can be used to periodically print the information to SYS\$OUTPUT in a batch job.

The /PROGRESS_REPORT qualifier will default to 60 seconds.

Example to display backup performance every 10 seconds:

```
$ RMU/BACKUP $1$DGA10:[DB]SAMPLE $1$DGA20:[BCK]SAMPLE /DISK /PROGRESS_REPORT=10
Read   18 MB ( 0%) at 18 MB/s, estimated completion time 14:10:41.15
.
.
.
Read  3934 MB (99%) at 28 MB/s, estimated completion time 14:10:39.86
```

- ◆ Read n MB = raw blocks read so far
- ◆ (n%) = percent of total blocks read
- ◆ n MB/s = transfer rate since last display
- ◆ estimated completion time = recalculated using the current transfer rate

For parallel backups with the /PROGRESS_REPORT qualifier, each worker process puts out its own progress report as in the following example:

```
WORKER_001: Read   41 MB (25%) at 41 MB/s, estimated completion time 14:55:32.96
WORKER_002: Read   37 MB (25%) at 37 MB/s, estimated completion time 14:55:32.96
WORKER_001: Read   75 MB (46%) at 34 MB/s, estimated completion time 14:55:33.62
WORKER_002: Read   69 MB (47%) at 31 MB/s, estimated completion time 14:55:33.57
WORKER_001: Read  104 MB (65%) at 29 MB/s, estimated completion time 14:55:33.96
WORKER_002: Read  100 MB (67%) at 30 MB/s, estimated completion time 14:55:33.62
WORKER_001: Read  135 MB (84%) at 30 MB/s, estimated completion time 14:55:33.92
WORKER_002: Read  130 MB (88%) at 30 MB/s, estimated completion time 14:55:33.66
```

The following restrictions currently exist:

- ◆ Restores from other than disk files do not display the percentage completed nor the estimated completion time:

```
$ RMU/RESTORE/NOCD $1$MGA500: SAMPLE, $1$MGA600: /REWIND /PROG=10
Read   72 MB at 14 MB/s
Read  135 MB at 15 MB/s
.
.
.
Read  1441 MB at 12 MB/s
```

13.1.28 /[NO]SNAPSHOTS, /[NO]DATA_FILE Added to RMU/MOVE_AREA

Currently, RMU/MOVE_AREA moves or creates a new version of BOTH the storage area data (*.RDA) and snapshot (*.SNP) files. This new syntax allows moving ONLY the data area file or ONLY the snapshot area file for all or for named storage areas. /NODATA_FILE and /NOSNAPSHOTS are positional qualifiers that can be specified globally as a default and/or for one or more named storage areas. They can be specified on the command line or in an options file using the existing RMU/MOVE_AREA /OPTION=filespec qualifier.

The syntax for these qualifiers is as follows:

```
/[NO]SNAPSHOTS[=( [FILE=filespec] , [ALLOCATION=n] )]
```

NOSNAPSHOTS does not move the storage area snapshot file(s). It only moves the data storage area file(s). SNAPSHOTS is the default. [=([FILE=filespec] , [ALLOCATION=n])] cannot be specified with NOSNAPSHOTS. SNAPSHOTS[=([FILE=filespec] , [ALLOCATION=n])] is an existing qualifier but now it can be negated. SNAPSHOTS as a local qualifier can override NOSNAPSHOTS as a global qualifier. NOSNAPSHOTS as a local qualifier can override SNAPSHOTS as a global qualifier.

```
/[NO]DATA_FILE
```

NODATA_FILE does not move the storage area data file(s). It only moves the snapshot storage area file(s). DATA_FILE is the default. It does not accept any values. DATA_FILE as a local qualifier can override NODATA_FILE as a global qualifier. NODATA_FILE as a local qualifier can override DATA_FILE as a global qualifier.

If NODATA_FILE is specified, the storage area data file is not moved or modified. A new version of the file will not be created. If NOSNAPSHOT is specified, the storage area snapshot file is not moved or modified. A new version of the file will not be created. Any existing RMU/MOVE_AREA qualifiers that would require an update/change to the data area file are disallowed if /NODATA_FILE is specified and any qualifiers that would require an update/change to the snapshot area file are disallowed if /NOSNAPSHOTS is specified.

Therefore, the following existing /MOVE_AREA qualifiers cannot be specified with either /NODATA_FILE or /NOSNAPSHOTS.

- ◆ /ROOT
- ◆ /BLOCKS_PER_PAGE
- ◆ /NODES_MAX
- ◆ /USERS_MAX

The following existing /MOVE_AREA qualifiers cannot be specified with /NODATA_FILE.

- ◆ /FILE
- ◆ /SPAMS
- ◆ /THRESHOLDS
- ◆ /READ_ONLY
- ◆ /READ_WRITE
- ◆ /EXTENSION

Oracle® Rdb for OpenVMS

The following existing /MOVE_AREA qualifiers cannot be specified with /NOSNAPSHOTS.

- ◆ /SNAPSHOTS=(FILE=filespec)
- ◆ /SNAPSHOTS=(ALLOCATION=n)

In the following example, only the storage area snapshot files are moved for all database storage areas.

```
$ RMU/MOVE_AREA/ALL/NODATA_FILE/NOLOG/DIR=[.MOVE] MF_PERSONNEL.RDB
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
```

In the following example, only the storage area data files are moved for all database storage areas.

```
$ RMU/MOVE_AREA/ALL/NOSNAPSHOTS/NOLOG/DIR=[.MOVE] MF_PERSONNEL.RDB
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
```

In the following example, only the snapshot storage area file is moved for the EMP_INFO storage area and only the data storage area file is moved for the JOBS storage area. Note that for the JOBS storage area, /DATA_FILE did not need to be specified since it is the default.

```
$ RMU/MOVE_AREA/NOLOG MF_PERSONNEL.RDB -
EMP_INFO /nodata_file -
           /snapshots=(file=DISK:[DIRECTORY]test_emp_info.snp), -
JOBS /data_file -
      /file=DISK:[DIRECTORY]test_jobs -
      /nosnapshots
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
```

In the following example, an options file is used to specify the storage areas to be moved. Only the data storage area file is moved for EMP_INFO; only the snapshot storage area file is moved for JOBS; and both the snapshot and data storage area files are moved for DEPARTMENTS. NOTE that /DATA_FILE and /SNAPSHOT are the defaults.

```
$ RMU/MOVE_AREA/NOLOG/DIR=DISK:[DIRECTORY]/OPTION=TESTMORE.OPT -
MF_PERSONNEL.RDB
EMP_INFO -
           /file=DISK:[DIRECTORY]test_emp_info.rda -
           /nosnapshot -
JOBS /nodata_file -
      /snapshot = (file=DISK:[DIRECTORY]test_jobs.snp)
DEPARTMENTS -
           /file=DISK:[DIRECTORY]test_departments -
           /snapshot = (file=DISK:[DIRECTORY]test_departments.snp)
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
```

In the following example, the global default qualifiers designate that only the snapshot files should be moved for all storage areas. However, an options file is used to override the default for specific storage areas. Therefore, only the data storage area file is moved for EMP_INFO, only the snapshot storage area file is moved for JOBS, and both the snapshot and data storage area files are moved for DEPARTMENTS. Note that, in this case, /DATA_FILE needed to be specified in the options file to override the global specification of /NODATA_FILE but /NODATA_FILE did not have to be specified in the options file. Also /NOSNAPSHOT had to be specified in the options file to override the assumed global default of /SNAPSHOT.

```
$ RMU/MOVE_AREA/ALL/DIR=DISK:[DIRECTORY]/NOLOG/NODATA_FILE-
/OPTION=TESTMOVE.OPT MF_PERSONNEL.RDB
```

```

EMP_INFO /data_file -
         /file=DISK:[DIRECTORY]test_emp_info.rda -
         /nosnapshot
JOBS     /nodata_file -
         /snapshot = (file=DISK:[DIRECTORY]test_jobs.snp)
DEPARTMENTS -
         /data_file -
         /file=DISK:[DIRECTORY]test_departments -
         /snapshot = (file=DISK:[DIRECTORY]test_departments.snp)
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
    
```

13.1.29 Enhancements for Compression Support in SQL EXPORT DATABASE Command

This release of Oracle Rdb introduces support for compression to the SQL EXPORT DATABASE statement and associated decompression to the SQL IMPORT DATABASE statement.

Data compression is applied to the user data exported to the internal (interchange) format file. Table rows, null byte vector and LIST OF BYTE VARYING data is compressed using either the LZW (Lempel–Ziv–Welch) technique or the ZLIB algorithm developed by Jean–loup Gailly and Mark Adler. Table metadata (column names and attributes) are never compressed and the resulting file remains a structured interchange file.

In past releases, it was possible that table data, stored in the database with compression enabled, would be many times smaller in the database than when exported by SQL. In the database, a simple and fast RLE (run–length encoding) algorithm is used to store rows but this data is fully expanded by the EXPORT DATABASE statement. Enabling compression allows the result data file to be more compact using less disk space and permitting faster network transmission. The tradeoff is that more CPU time will be required for the compression and decompression of the data.

Changes to the SQL EXPORT DATABASE Statement

A new COMPRESSION clause has been added to the SQL EXPORT DATABASE statement. The default remains NO COMPRESSION. This clause accepts the following optional keywords: LZW, and ZLIB. The compression algorithms used are ZLIB (the default) or LZW. ZLIB allows further tuning with the LEVEL option that accepts a numeric level between 1 and 9. The default of 6 is usually a good trade off between result file size and the CPU cost of the compression.

```

--> NO COMPRESSION -----+>
|
|--> COMPRESSION -----+
|
|   |--> USING |--> LZW -----+
|               |
|               |--> ZLIB -----+
|                   |
|                   |--> ( LEVEL numeric-literal ) --+
    
```

The following example shows the specification of the COMPRESSION clause.

```
SQL> export database
```



```

cont> filename COMPLETE_WORKS
cont> into COMPLETE_WORKS.RBR
cont> compression using ZLIB (level 9)
cont> ;

```

Changes to the IMPORT DATABASE Statement

The metadata in the interchange file defines the compression algorithm to be used by the IMPORT DATABASE statement and indicates which tables were compressed by the EXPORT DATABASE statement.

Usage Notes

- ◆ Only the user data is compressed, therefore, additional compression may be applied using various third party compression tools such as ZIP. It is not the goal of SQL to replace such tools.
- ◆ Only one of LZW or ZLIB may be specified for the COMPRESSION option. The LEVEL clause may not be used with LZW compression technique.
- ◆ The generated interchange file (.rbr) can be processed using the RMU Dump Export command.
- ◆ The EXPORT DATABASE statement uses compression in multiple streams. Each table is treated as a separate compression stream as is each table's null byte vector and LIST OF BYTE VARYING columns.
- ◆ In some cases, compression may automatically be disabled. When the null byte vector or row data is small (less than 9 octets), the compression overhead would typically grow the data.

13.1.30 /[NO]PARTITIONS Qualifier Added to RMU/ANALYZE/INDEXES

A new /PARTITIONS qualifier has been added to RMU/ANALYZE/INDEXES which allows data to be collected and output for individual index partitions for sorted, sorted ranked and hashed indexes which are partitioned across multiple Oracle Rdb database storage areas. Previously, only index wide data was displayed by RMU/ANALYZE/INDEXES whether or not an index was partitioned.

The new /PARTITIONS qualifier will work with existing RMU/ANALYZE/INDEXES qualifiers. RMU/ANALYZE/INDEXES/PARTITIONS can be used with /OPTIONS=FULL and /OPTIONS=DEBUG to include data for different numbered levels of individual index partitions. RMU/ANALYZE/INDEXES/PARTITIONS can be used with /BINARY_OUTPUT to output record definition (*.RRD) and unload files (*.UNL) that can be used to load partition data records into an Oracle Rdb database table using the RMU/LOAD command.

If /PARTITIONS is not specified or if /PARTITIONS is specified and an index is not partitioned, only index wide data will be output by RMU/ANALYZE/INDEXES.

The syntax for this new qualifier is as follows:

```
/[NO]PARTITIONS
```

/NOPARTITIONS is the default.

Oracle® Rdb for OpenVMS

The following example shows that only index wide data is output for index I1 in database PART_IND_DB.RDB if /PARTITIONS is not specified.

```
$ RMU/ANALYZE/INDEX PART_IND_DB I1
```

```
-----  
Indices for database - DISK:[DIRECTORY]PART_IND_DB.RDB;  
Created dd-mmm-yyyy hh:mm:ss.xxxx
```

```
-----  
Index I1 for relation T1 duplicates allowed  
Max Level: 3, Nodes: 898, Used/Avail: 194731/357404 (54%), Keys: 10892,  
Records: 10000  
Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0  
-----
```

The following example shows that data for each index partition is output after the index wide data for index I1 in database PART_IND_DB.RDB if /PARTITIONS is specified. The partition names are P1 through P6 and the storage area names for the partitions are INDEX1 through INDEX6. The partition data is sorted by partition name.

```
$ RMU/ANALYZE/INDEX/PARTITION PART_IND_DB I1
```

```
-----  
Indices for database - DISK:[DIRECTORY]PART_IND_DB.RDB;  
Created dd-mmm-yyyy hh:mm:ss.xxxx
```

```
-----  
Index I1 for relation T1 duplicates allowed  
Max Level: 3, Nodes: 898, Used/Avail: 194731/357404 (54%), Keys: 10892,  
Records: 10000  
Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0
```

```
Partition P1 in area INDEX1  
Max Level: 1, Nodes: 1, Used/Avail: 0/398 (0%), Keys: 0, Records: 0  
Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0
```

```
Partition P2 in area INDEX2  
Max Level: 1, Nodes: 1, Used/Avail: 21/398 (5%), Keys: 1, Records: 1  
Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0
```

```
Partition P3 in area INDEX3  
Max Level: 1, Nodes: 1, Used/Avail: 72/398 (18%), Keys: 4, Records: 4  
Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0
```

```
Partition P4 in area INDEX4  
Max Level: 2, Nodes: 9, Used/Avail: 1789/3582 (49%), Keys: 103, Records: 95  
Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0
```

```
Partition P5 in area INDEX5  
Max Level: 3, Nodes: 885, Used/Avail: 192849/352230 (54%), Keys: 10784,  
Records: 9900  
Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0
```

```
Partition P6 in area INDEX6  
Max Level: 1, Nodes: 1, Used/Avail: 0/398 (0%), Keys: 0, Records: 0  
Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0  
-----
```

Oracle® Rdb for OpenVMS

The following example shows that data for each level within each index partition is output after the index wide data for index I1 in database PART_IND_DB.RDB if /OPTION=FULL is specified. The partition names are P1 through P6 and the storage area names for the partitions are INDEX1 through INDEX6. The levels are numbered in descending order.

```
$ RMU/ANALYZE/INDEX/PARTITION/OPTION=FULL PART_IND_DB I1
```

```
-----  
Indices for database - DISK:[DIRECTORY]PART_IND_DB.RDB;  
Created dd-mmm-yyyy hh:mm:ss.xxxx  
-----
```

```
Index I1 for relation T1 duplicates allowed
```

```
Max Level: 3, Nodes: 898, Used/Avail: 194731/357404 (54%), Keys: 10892,  
Records: 10000  
Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0
```

```
Level: 3, Nodes: 1, Used/Avail: 184/398 (46%), Keys: 23, Records: 0  
Level: 2, Nodes: 24, Used/Avail: 6197/9552 (64%), Keys: 869, Records: 0  
Level: 1, Nodes: 873, Used/Avail: 188350/347454 (54%), Keys: 10000,  
Records: 10000
```

```
Partition P1 in area INDEX1
```

```
Max Level: 1, Nodes: 1, Used/Avail: 0/398 (0%), Keys: 0, Records: 0  
Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0
```

```
Level: 1, Nodes: 1, Used/Avail: 0/398 (0%), Keys: 0, Records: 0
```

```
Partition P2 in area INDEX2
```

```
Max Level: 1, Nodes: 1, Used/Avail: 21/398 (5%), Keys: 1, Records: 1  
Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0
```

```
Level: 1, Nodes: 1, Used/Avail: 21/398 (5%), Keys: 1, Records: 1
```

```
Partition P3 in area INDEX3
```

```
Max Level: 1, Nodes: 1, Used/Avail: 72/398 (18%), Keys: 4, Records: 4  
Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0
```

```
Level: 1, Nodes: 1, Used/Avail: 72/398 (18%), Keys: 4, Records: 4
```

```
Partition P4 in area INDEX4
```

```
Max Level: 2, Nodes: 9, Used/Avail: 1789/3582 (49%), Keys: 103, Records: 95  
Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0
```

```
Level: 2, Nodes: 1, Used/Avail: 56/398 (14%), Keys: 8, Records: 0  
Level: 1, Nodes: 8, Used/Avail: 1733/3184 (54%), Keys: 95, Records: 95
```

```
Partition P5 in area INDEX5
```

```
Max Level: 3, Nodes: 885, Used/Avail: 192849/352230 (54%), Keys: 10784,  
Records: 9900  
Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0
```

```
Level: 3, Nodes: 1, Used/Avail: 184/398 (46%), Keys: 23, Records: 0  
Level: 2, Nodes: 23, Used/Avail: 6141/9154 (67%), Keys: 861, Records: 0  
Level: 1, Nodes: 861, Used/Avail: 186524/342678 (54%), Keys: 9900,  
Records: 9900
```

```
Partition P6 in area INDEX6
```

```
Max Level: 1, Nodes: 1, Used/Avail: 0/398 (0%), Keys: 0, Records: 0  
Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0
```

Oracle® Rdb for OpenVMS

Level: 1, Nodes: 1, Used/Avail: 0/398 (0%), Keys: 0, Records: 0

The following example shows that, if /BINARY_OUTPUT is specified, a PARTIND.UNL file and a PARTIND.RRD file are created that can be used with the RMU/LOAD command to load partition data for the index I1 into an Oracle Rdb database table. The PARTIND.RRD file contains the fields RMU\$PARTITION_NAME and RMU\$AREA_NAME which will contain the partition name and area name for each record in the PARTIND.UNL file which contains partition specific data.

```
$ RMU/ANALYZE/INDEX/PARTITION/BINARY=(FILE=PARTIND.UNL,RECORD=PARTIND.RRD) -
/OUTPUT=I1.OUT PART_IND_DB I1
$ TYPE PARTIND.RRD
DEFINE FIELD RMU$DATE DATATYPE IS DATE.
DEFINE FIELD RMU$INDEX_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$RELATION_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$PARTITION_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$AREA_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$LEVEL DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$FLAGS DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$USED DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$AVAILABLE DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_MAP DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_USED DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_AVAILABLE DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$KEY_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DATA_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_KEY_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_DATA_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_COMPRESSED_IKEY_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_IKEY_COUNT DATATYPE IS F_FLOATING.
DEFINE RECORD RMU$ANALYZE_INDEX.
    RMU$DATE.
    RMU$INDEX_NAME.
    RMU$RELATION_NAME.
    RMU$PARTITION_NAME.
    RMU$AREA_NAME.
    RMU$LEVEL.
    RMU$FLAGS.
    RMU$COUNT.
    RMU$USED.
    RMU$AVAILABLE.
    RMU$DUPLICATE_COUNT.
    RMU$DUPLICATE_MAP.
    RMU$DUPLICATE_USED.
    RMU$DUPLICATE_AVAILABLE.
    RMU$KEY_COUNT.
    RMU$DATA_COUNT.
    RMU$DUPLICATE_KEY_COUNT.
    RMU$DUPLICATE_DATA_COUNT.
    RMU$TOTAL_COMPRESSED_IKEY_COUNT.
    RMU$TOTAL_IKEY_COUNT.
END RMU$ANALYZE_INDEX RECORD.
```

13.1.31 /[NO]PARTITIONS Qualifier Added to RMU/ANALYZE Storage Statistics

A new /[NO]PARTITIONS[=(TABLES,INDEXES)] qualifier has been added to RMU/ANALYZE storage statistics which allows data to be collected and output for individual table and/or index partitions across multiple Oracle Rdb database storage areas. Previously, only statistics for storage areas and the logical areas contained within each storage area could be displayed. Now, if /PARTITIONS is specified, first statistics for each Oracle Rdb database storage area containing partitions is output. Then statistics for each partition logical area defined for each partitioned table is output. Finally, statistics for each partition logical area defined for each partitioned index is output.

The syntax for this new qualifier is as follows:

```
/[NO]PARTITIONS[=(TABLES,INDEXES)]
```

- ◆ /NOPARTITIONS is the default.
- ◆ /PARTITIONS=TABLES only outputs partitioned table statistics.
- ◆ /PARTITIONS=INDEXES only outputs partitioned index statistics.
- ◆ If only /PARTITIONS is specified, both partitioned table and partitioned index statistics are output.

If /PARTITIONS=TABLES is specified, only statistics for partitioned tables are output. If /PARTITIONS=INDEXES is specified, only statistics for partitioned indexes are output. If the /LAREAS qualifier is used to specify a list of names of partitioned tables and/or partitioned indexes, only statistics for those tables and/or indexes will be output. If the /LAREAS qualifier is used to specify a list of logical area identifier numbers, only those logical area partitions for partitioned tables and/or indexes will be output. IF /BINARY_OUTPUT is specified with /PARTITIONS, record definition (*.RRD) and binary unload files (*.UNL) are created that can be used to load storage area and logical area data records for partitioned tables and/or indexes into an Oracle Rdb database table using the RMU/LOAD command.

The RMU/ANALYZE/PARTITIONS qualifier for storage statistics cannot be specified with the following RMU/ANALYZE qualifiers: /PLACEMENT, /CARDINALITY, /INDEXES, /AREAS, /START, /END and /EXCLUDE. Note that a new qualifier not discussed here has been added to RMU/ANALYZE/INDEXES with the syntax /[NO]PARTITIONS for index specific statistics (see previous topic). If /LAREAS is used, it must specify a partitioned table name or index name or a logical area identifier number for a logical area defined for a partitioned table or partitioned index.

In the following example, for the PART_DB database with one partitioned table T1 and one partitioned index I1, first statistics for the storage areas containing table and index partitions are output: DATA1.RDA, INDEX1.RDA and INDEX2.RDA. Then statistics for the partitioned table T1, defined with one partition SYS_P00059 in area DATA1, is output. Then statistics for the two partitioned index I1 partitions, P1 in the INDEX1 storage area and P2 in the INDEX2 storage area, are output. Note that if /PARTITIONS=TABLES was specified for the PART_DB database, only statistics for the partitioned table T1 would be output and if /PARTITIONS=INDEXES was specified, only statistics for the partitioned index I1 would be output.

```
$ RMU/ANALYZE/PARTITIONS PART_DB
```

```
Areas containing partitions for database - DISK:[DIRECTORY]PART_DB.RDB;1
Created 9-JUN-2012 14:11:38.43
```

Oracle® Rdb for OpenVMS

Storage analysis for storage area: DATA1 - file: DISK:[DIRECTORY]DATA1.RDA;1
Area_id: 2, Page length: 1024, Last page: 703

Bytes free: 366675 (51%), bytes overhead: 164457 (23%)
Spam count: 1, AIP count: 0, ABM count: 3
Data records: 10000, bytes used: 188740 (26%)
 average length: 19, compression ratio: 0.90
 index records: 0, bytes used: 0 (0%)

Storage analysis for storage area: INDEX1 - file: DISK:[DIRECTORY]INDEX1.RDA;1
Area_id: 3, Page length: 1024, Last page: 703

Bytes free: 685993 (95%), bytes overhead: 33451 (5%)
Spam count: 1, AIP count: 0, ABM count: 3
Data records: 1, bytes used: 428 (0%)
 average length: 428, compression ratio: 1.00
 index records: 1, bytes used: 428 (0%)
 B-Tree: 428, Hash: 0, Duplicate: 0, Overflow: 0

Storage analysis for storage area: INDEX2 - file: DISK:[DIRECTORY]INDEX2.RDA;1
Area_id: 4, Page length: 1024, Last page: 703

Bytes free: 685993 (95%), bytes overhead: 33451 (5%)
Spam count: 1, AIP count: 0, ABM count: 3
Data records: 1, bytes used: 428 (0%)
 average length: 428, compression ratio: 1.00
 index records: 1, bytes used: 428 (0%)
 B-Tree: 428, Hash: 0, Duplicate: 0, Overflow: 0

Partitioned Tables for database - DISK:[DIRECTORY]PART_DB.RDB;1
Created 9-JUN-2012 14:11:38.43

Storage analysis for Partitioned Table: T1

Partition SYS_P00059 in area DATA1
Logical area: T1 Logical area id : 59

Larea id: 59, Record type: 31, Record length: 26, Compressed

Data records: 10000, bytes used: 188740 (26%)
 average length: 19, compression ratio: 0.90

Partitioned Indexes for database - DISK:[DIRECTORY]PART_DB.RDB;1
Created 9-JUN-2012 14:11:38.43

Oracle® Rdb for OpenVMS

Storage analysis for Partitioned Index: I1

Partition P1 in area INDEX1
Logical area: I1 Logical area id : 60

Larea id: 60, Record type: 0, Record length: 215, Not Compressed

Data records: 1, bytes used: 428 (0%)
average length: 428

Partition P2 in area INDEX2
Logical area: I1 Logical area id : 61

Larea id: 61, Record type: 0, Record length: 215, Not Compressed

Data records: 1, bytes used: 428 (0%)
average length: 428

In the following example for the PART_DB database with one partitioned table T1 and one partitioned index I1, the /LAREA qualifier is specified to name the index I1. Therefore only data for the partitioned index I1 is output.

```
$ RMU/ANALYZE/PARTITIONS=INDEXES/LAREA=I1 PART_DB
```

```
Areas containing partitions for database - DISK:[DIRECTORY]PART_DB.RDB;1  
Created 9-JUN-2013 14:17:47.13
```

Storage analysis for storage area: INDEX1 - file: DISK:[DIRECTORY]INDEX1.RDA;1
Area_id: 3, Page length: 1024, Last page: 703

Bytes free: 685993 (95%), bytes overhead: 33451 (5%)
Spam count: 1, AIP count: 0, ABM count: 3
Data records: 1, bytes used: 428 (0%)
average length: 428, compression ratio: 1.00
index records: 1, bytes used: 428 (0%)
B-Tree: 428, Hash: 0, Duplicate: 0, Overflow: 0

Storage analysis for storage area: INDEX2 - file: DISK:[DIRECTORY]INDEX2.RDA;1
Area_id: 4, Page length: 1024, Last page: 703

Bytes free: 685993 (95%), bytes overhead: 33451 (5%)
Spam count: 1, AIP count: 0, ABM count: 3
Data records: 1, bytes used: 428 (0%)
average length: 428, compression ratio: 1.00
index records: 1, bytes used: 428 (0%)
B-Tree: 428, Hash: 0, Duplicate: 0, Overflow: 0

Oracle® Rdb for OpenVMS

```
-----  
-----  
Partitioned Indexes for database - DISK:[DIRECTORY]PART_DB.RDB;1  
Created 9-JUN-2012 14:17:47.13  
-----
```

```
Storage analysis for Partitioned Index: I1  
-----
```

```
Partition P1 in area INDEX1  
Logical area: I1 Logical area id : 60
```

```
Larea id: 60, Record type: 0, Record length: 215, Not Compressed
```

```
Data records: 1, bytes used: 428 (0%)  
average length: 428  
-----
```

```
Partition P2 in area INDEX2  
Logical area: I1 Logical area id : 61
```

```
Larea id: 61, Record type: 0, Record length: 215, Not Compressed
```

```
Data records: 1, bytes used: 428 (0%)  
average length: 428  
-----
```

In the following example for the PART_DB database with one partitioned table T1 and one partitioned index I1, the /LAREA qualifier specifies the logical area identifier number "60". Therefore only data for the single partition P1 for the index I1 is output.

```
$ RMU/ANALYZE/PARTITIONS/LAREA=60 PART_DB.RDB
```

```
Areas containing partitions for database - DISK:[DIRECTORY]PART_DB.RDB;1  
Created 9-JUN-2012 15:25:23.68  
-----
```

```
Storage analysis for storage area: INDEX1 - file: DISK:[DIRECTORY]INDEX1.RDA;1  
Area_id: 3, Page length: 1024, Last page: 703
```

```
Bytes free: 685993 (95%), bytes overhead: 33451 (5%)  
Spam count: 1, AIP count: 0, ABM count: 3  
Data records: 1, bytes used: 428 (0%)  
average length: 428, compression ratio: 1.00  
index records: 1, bytes used: 428 (0%)  
B-Tree: 428, Hash: 0, Duplicate: 0, Overflow: 0  
-----  
-----
```

```
Partitioned Indexes for database - DISK:[DIRECTORY]PART_DB.RDB;1  
Created 9-JUN-2012 15:25:23.68  
-----
```


Oracle® Rdb for OpenVMS

Storage analysis for Partitioned Index: I1

```
-----  
Partition P1 in area INDEX1  
Logical area: I1 Logical area id : 60  
  
Larea id: 60, Record type: 0, Record length: 215, Not Compressed  
  
Data records: 1, bytes used: 428 (0%)  
  average length: 428  
-----
```

The following example shows that if /BINARY_OUTPUT is specified, a PART.UNL file and a PART.RRD file are created that can be used with the RMU/LOAD command to load storage partition data into an Oracle Rdb database table. The PART.RRD file contains the new fields RMU\$TABLE_NAME, RMU\$INDEX_NAME, RMU\$PARTITION_NAME and RMU\$ST_AREA_NAME for partition specific data. Note that RMU\$AREA_NAME contains the logical area name and RMU\$ST_AREA_NAME contains the storage area name that contains the partition or index.

```
$ RMU/ANALYZE/PARTITIONS -  
  -$ /BINARY=(FILE=PART.UNL,RECORD=PART.RRD) -  
  -$ /OUTPUT=PART.OUT PART_DB  
$ TYPE PART.RRD  
DEFINE FIELD RMU$DATE DATATYPE IS DATE.  
DEFINE FIELD RMU$AREA_NAME DATATYPE IS TEXT SIZE IS 32.  
DEFINE FIELD RMU$TABLE_NAME DATATYPE IS TEXT SIZE IS 32.  
DEFINE FIELD RMU$INDEX_NAME DATATYPE IS TEXT SIZE IS 32.  
DEFINE FIELD RMU$PARTITION_NAME DATATYPE IS TEXT SIZE IS 32.  
DEFINE FIELD RMU$ST_AREA_NAME DATATYPE IS TEXT SIZE IS 32.  
DEFINE FIELD RMU$STORAGE_AREA_ID DATATYPE IS SIGNED WORD.  
DEFINE FIELD RMU$FLAGS DATATYPE IS SIGNED WORD.  
DEFINE FIELD RMU$TOTAL_BYTES DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$EXPANDED_BYTES DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$FRAGMENTED_BYTES DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$EXPANDED_FRAGMENT_BYTES DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$TOTAL_COUNT DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$FRAGMENTED_COUNT DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$FRAGMENT_COUNT DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$PAGE_LENGTH DATATYPE IS SIGNED WORD.  
DEFINE FIELD RMU$MAX_PAGE_NUMBER DATATYPE IS SIGNED LONGWORD.  
DEFINE FIELD RMU$FREE_BYTES DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$OVERHEAD_BYTES DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$AIP_COUNT DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$ABM_COUNT DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$SPAM_COUNT DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$INDEX_COUNT DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$BTREE_NODE_BYTES DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$HASH_BYTES DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$DUPLICATES_BYTES DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$OVERFLOW_BYTES DATATYPE IS F_FLOATING.  
DEFINE FIELD RMU$LOGICAL_AREA_ID DATATYPE IS SIGNED WORD.  
DEFINE FIELD RMU$RELATION_ID DATATYPE IS SIGNED WORD.  
DEFINE FIELD RMU$RECORD_ALLOCATION_SIZE DATATYPE IS SIGNED WORD.  
DEFINE FIELD RMU$TOTAL_SPACE DATATYPE IS F_FLOATING.  
DEFINE RECORD RMU$ANALYZE_AREA.  
  RMU$DATE.  
  RMU$AREA_NAME.
```

```

RMU$TABLE_NAME .
RMU$INDEX_NAME .
RMU$PARTITION_NAME .
RMU$ST_AREA_NAME .
RMU$STORAGE_AREA_ID .
RMU$FLAGS .
RMU$TOTAL_BYTES .
RMU$EXPANDED_BYTES .
RMU$FRAGMENTED_BYTES .
RMU$EXPANDED_FRAGMENT_BYTES .
RMU$TOTAL_COUNT .
RMU$FRAGMENTED_COUNT .
RMU$FRAGMENT_COUNT .
RMU$PAGE_LENGTH .
RMU$MAX_PAGE_NUMBER .
RMU$FREE_BYTES .
RMU$OVERHEAD_BYTES .
RMU$AIP_COUNT .
RMU$ABM_COUNT .
RMU$SPAM_COUNT .
RMU$INDEX_COUNT .
RMU$BTREE_NODE_BYTES .
RMU$HASH_BYTES .
RMU$DUPLICATES_BYTES .
RMU$OVERFLOW_BYTES .
RMU$LOGICAL_AREA_ID .
RMU$RELATION_ID .
RMU$RECORD_ALLOCATION_SIZE .
RMU$TOTAL_SPACE .
END RMU$ANALYZE_AREA RECORD .

```

13.1.32 /[NO]PARTITIONS Qualifier Added to RMU/ANALYZE/PLACEMENT

The RMU/ANALYZE/PLACEMENT command collects and displays statistical information describing table row placement relative to the index structure for an Oracle Rdb database. A new /PARTITIONS qualifier has been added to RMU/ANALYZE/PLACEMENT which allows placement data to be collected and output for individual index partitions for sorted, sorted ranked and hashed indexes which are partitioned across multiple Oracle Rdb database storage areas. Previously, only index wide data was displayed by RMU/ANALYZE/PLACEMENT whether or not an index was partitioned. The new /PARTITIONS qualifier will work with existing RMU/ANALYZE/PLACEMENT qualifiers.

RMU/ANALYZE/PLACEMENT/PARTITIONS can be used with /OPTIONS=FULL and /OPTIONS=DEBUG to include histogram displays for individual index partitions.

RMU/ANALYZE/PLACEMENT/PARTITIONS can be used with /BINARY_OUTPUT to output record definition (*.RRD) and unload files (*.UNL) that can be used to load partition placement data records into an Oracle Rdb database table using the RMU/LOAD command.

The syntax for this new qualifier is as follows:

```
/[NO]PARTITIONS
```

/NOPARTITIONS is the default.

Oracle® Rdb for OpenVMS

If /PARTITIONS is not specified or if /PARTITIONS is specified and an index is not partitioned, only index wide placement data will be output by RMU/ANALYZE/PLACEMENT.

The following example shows that only index wide placement data is output for index I1 in database PART_IND_DB.RDB, if /PARTITIONS is not specified.

```
$ RMU/ANALYZE/PLACEMENT PART_IND_DB I1
```

```
-----  
Indices for database - DISK:[DIRECTORY]PART_IND_DB.RDB;  
Created dd-mmm-yyyy hh:mm:ss.xxxx
```

```
-----  
Index I1 for relation T1 duplicates allowed  
Levels: 3, Nodes: 898, Keys: 10892, Records: 10000  
  Dup nodes: 0, Dup keys: 0, Dup records: 0  
Maximum path length -- dbkeys: 4, IO range: 2 to 4  
Average path length -- dbkeys: 3.99, IO range: 3.96 to 3.96
```

The following example shows that placement data for each index partition is output after the index wide placement data for index I1 in database PART_IND_DB.RDB, if /PARTITIONS is specified. The partition names are P1 through P6 and the storage area names for the partitions are INDEX1 through INDEX6. The partition data is sorted by partition name.

```
$ RMU/ANALYZE/PLACEMENT/PARTITION PART_IND_DB I1
```

```
-----  
Indices for database - DISK:[DIRECTORY]PART_IND_DB.RDB;  
Created dd-mmm-yyyy hh:mm:ss.xxxx
```

```
-----  
Index I1 for relation T1 duplicates allowed  
Levels: 3, Nodes: 898, Keys: 10892, Records: 10000  
  Dup nodes: 0, Dup keys: 0, Dup records: 0  
Maximum path length -- dbkeys: 4, IO range: 2 to 4  
Average path length -- dbkeys: 3.99, IO range: 3.96 to 3.96
```

```
Partition P1 in area INDEX1  
Levels: 1, Nodes: 1, Keys: 0, Records: 0  
  Dup nodes: 0, Dup keys: 0, Dup records: 0  
Maximum path length -- dbkeys: 0, IO range: 0 to 0  
Average path length -- dbkeys: 0.00, IO range: 0.00 to 0.00
```

```
Partition P2 in area INDEX2  
Levels: 1, Nodes: 1, Keys: 1, Records: 1  
  Dup nodes: 0, Dup keys: 0, Dup records: 0  
Maximum path length -- dbkeys: 2, IO range: 2 to 2  
Average path length -- dbkeys: 2.00, IO range: 2.00 to 2.00
```

```
Partition P3 in area INDEX3  
Levels: 1, Nodes: 1, Keys: 4, Records: 4  
  Dup nodes: 0, Dup keys: 0, Dup records: 0  
Maximum path length -- dbkeys: 2, IO range: 2 to 2  
Average path length -- dbkeys: 2.00, IO range: 2.00 to 2.00
```

Oracle® Rdb for OpenVMS

Partition P4 in area INDEX4
Levels: 2, Nodes: 9, Keys: 103, Records: 95
Dup nodes: 0, Dup keys: 0, Dup records: 0
Maximum path length -- dbkeys: 3, IO range: 2 to 3
Average path length -- dbkeys: 3.00, IO range: 2.87 to 2.87

Partition P5 in area INDEX5
Levels: 3, Nodes: 885, Keys: 10784, Records: 9900
Dup nodes: 0, Dup keys: 0, Dup records: 0
Maximum path length -- dbkeys: 4, IO range: 3 to 4
Average path length -- dbkeys: 4.00, IO range: 3.97 to 3.97

Partition P6 in area INDEX6
Levels: 1, Nodes: 1, Keys: 0, Records: 0
Dup nodes: 0, Dup keys: 0, Dup records: 0
Maximum path length -- dbkeys: 0, IO range: 0 to 0
Average path length -- dbkeys: 0.00, IO range: 0.00 to 0.00

The following example shows that placement data and histograms for each index partition are output after the index wide placement data and histograms for index I1 in database PART_IND_DB.RDB, if /OPTION=FULL is specified. The partition names are P1 through P6 and the storage area names for the partitions are INDEX1 through INDEX6.

```
$ RMU/ANALYZE/PLACEMENT/PARTITION/OPTION=FULL PART_IND_DB I1
```

```
Indices for database - DISK:[DIRECTORY]PART_IND_DB.RDB;  
Created dd-mmm-yyyy hh:mm:ss.xxxx
```

```
Index I1 for relation T1 duplicates allowed  
Levels: 3, Nodes: 898, Keys: 10892, Records: 10000  
Dup nodes: 0, Dup keys: 0, Dup records: 0  
Maximum path length -- dbkeys: 4, IO range: 2 to 4  
Average path length -- dbkeys: 3.99, IO range: 3.96 to 3.96
```

dbkey path length vs. frequency

```
6 | (0)  
5 | (0)  
4 | ===== (9900)  
3 | (95)  
2 | (5)  
1 | (0)
```

MAX IO path length vs. frequency

```
6 | (0)  
5 | (0)  
4 | ===== (9624)  
3 | == (359)  
2 | (17)  
1 | (0)
```

Oracle® Rdb for OpenVMS

MIN IO path length vs. frequency

```
6 | (0)
5 | (0)
4 |===== (9624)
3 |== (359)
2 | (17)
1 | (0)
```

Partition P1 in area INDEX1

Levels: 1, Nodes: 1, Keys: 0, Records: 0

Dup nodes: 0, Dup keys: 0, Dup records: 0

Maximum path length -- dbkeys: 0, IO range: 0 to 0

Average path length -- dbkeys: 0.00, IO range: 0.00 to 0.00

Partition P2 in area INDEX2

Levels: 1, Nodes: 1, Keys: 1, Records: 1

Dup nodes: 0, Dup keys: 0, Dup records: 0

Maximum path length -- dbkeys: 2, IO range: 2 to 2

Average path length -- dbkeys: 2.00, IO range: 2.00 to 2.00

dbkey path length vs. frequency

```
6 | (0)
5 | (0)
4 | (0)
3 | (0)
2 |===== (1)
1 | (0)
```

MAX IO path length vs. frequency

```
6 | (0)
5 | (0)
4 | (0)
3 | (0)
2 |===== (1)
1 | (0)
```

MIN IO path length vs. frequency

```
6 | (0)
5 | (0)
4 | (0)
3 | (0)
2 |===== (1)
1 | (0)
```

Partition P3 in area INDEX3

Levels: 1, Nodes: 1, Keys: 4, Records: 4

Dup nodes: 0, Dup keys: 0, Dup records: 0

Maximum path length -- dbkeys: 2, IO range: 2 to 2

Average path length -- dbkeys: 2.00, IO range: 2.00 to 2.00

Oracle® Rdb for OpenVMS

dbkey path length vs. frequency

```
6 | (0)
5 | (0)
4 | (0)
3 | (0)
2 | ===== (4)
1 | (0)
```

MAX IO path length vs. frequency

```
6 | (0)
5 | (0)
4 | (0)
3 | (0)
2 | ===== (4)
1 | (0)
```

MIN IO path length vs. frequency

```
6 | (0)
5 | (0)
4 | (0)
3 | (0)
2 | ===== (4)
1 | (0)
```

Partition P4 in area INDEX4

Levels: 2, Nodes: 9, Keys: 103, Records: 95

Dup nodes: 0, Dup keys: 0, Dup records: 0

Maximum path length -- dbkeys: 3, IO range: 2 to 3

Average path length -- dbkeys: 3.00, IO range: 2.87 to 2.87

dbkey path length vs. frequency

```
6 | (0)
5 | (0)
4 | (0)
3 | ===== (95)
2 | (0)
1 | (0)
```

MAX IO path length vs. frequency

```
6 | (0)
5 | (0)
4 | (0)
3 | ===== (83)
2 | ===== (12)
1 | (0)
```

Oracle® Rdb for OpenVMS

MIN IO path length vs. frequency

```
6 | (0)
5 | (0)
4 | (0)
3 | ===== (83)
2 | ===== (12)
1 | (0)
```

Partition P5 in area INDEX5

Levels: 3, Nodes: 885, Keys: 10784, Records: 9900

Dup nodes: 0, Dup keys: 0, Dup records: 0

Maximum path length -- dbkeys: 4, IO range: 3 to 4

Average path length -- dbkeys: 4.00, IO range: 3.97 to 3.97

dbkey path length vs. frequency

```
6 | (0)
5 | (0)
4 | ===== (9900)
3 | (0)
2 | (0)
1 | (0)
```

MAX IO path length vs. frequency

```
6 | (0)
5 | (0)
4 | ===== (9624)
3 | = (276)
2 | (0)
1 | (0)
```

MIN IO path length vs. frequency

```
6 | (0)
5 | (0)
4 | ===== (9624)
3 | = (276)
2 | (0)
1 | (0)
```

Partition P6 in area INDEX6

Levels: 1, Nodes: 1, Keys: 0, Records: 0

Dup nodes: 0, Dup keys: 0, Dup records: 0

Maximum path length -- dbkeys: 0, IO range: 0 to 0

Average path length -- dbkeys: 0.00, IO range: 0.00 to 0.00

The following example shows that, if /BINARY_OUTPUT is specified, a PARTIND.UNL file and a PARTIND.RRD file are created that can be used with the RMU/LOAD command to load partition placement data for the index I1 into an Oracle Rdb database table. The PARTIND.RRD file contains the fields RMU\$PARTITION_NAME and RMU\$AREA_NAME, which will contain the partition

name and area name for each record in the PARTIND.UNL file. PARTIND.UNL contains partition specific data.

```

$ RMU/ANALYZE/PARTITION/PARTITION/BINARY=(FILE=PARTIND.UNL,RECORD=PARTIND.RRD) -
/OUTPUT=I1.OUT PART_IND_DB I1
$ TYPE PARTIND.RRD
DEFINE FIELD RMU$DATE DATATYPE IS DATE.
DEFINE FIELD RMU$INDEX_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$RELATION_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$PARTITION_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$AREA_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$LEVEL DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$FLAGS DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_MAP_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$KEY_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_KEY_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DATA_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_DATA_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_KEY_PATH DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_PAGE_PATH DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_BUFFER_PATH DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$MAX_KEY_PATH DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$MAX_PAGE_PATH DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$MIN_BUF_PATH DATATYPE IS F_FLOATING.
DEFINE RECORD RMU$ANALYZE_PLACEMENT.
    RMU$DATE.
    RMU$INDEX_NAME.
    RMU$RELATION_NAME.
    RMU$PARTITION_NAME.
    RMU$AREA_NAME.
    RMU$LEVEL.
    RMU$FLAGS.
    RMU$COUNT.
    RMU$DUPLICATE_COUNT.
    RMU$DUPLICATE_MAP_COUNT.
    RMU$KEY_COUNT.
    RMU$DUPLICATE_KEY_COUNT.
    RMU$DATA_COUNT.
    RMU$DUPLICATE_DATA_COUNT.
    RMU$TOTAL_KEY_PATH.
    RMU$TOTAL_PAGE_PATH.
    RMU$TOTAL_BUFFER_PATH.
    RMU$MAX_KEY_PATH.
    RMU$MAX_PAGE_PATH.
    RMU$MIN_BUF_PATH.
END RMU$ANALYZE_PLACEMENT RECORD.

```

13.1.33 New RMU/[NO]ASSIST Qualifier for Commands Using Tape Drives

A new qualifier has been added to improve tape handling when using RMU commands which use tape volumes. The following commands have been enhanced with the new "[NO]ASSIST" qualifier:

- ◆ RMU/BACKUP
- ◆ RMU/BACKUP/AFTER_JOURNAL

- ◆ RMU/DUMP/BACKUP_FILE
- ◆ RMU/DUMP/AFTER_JOURNAL
- ◆ RMU/OPTIMIZE
- ◆ RMU/RECOVER
- ◆ RMU/RESTORE

This qualifier specifies where tape handling requests are to be sent. With 'NoAssist' these requests are sent to the current process's SYSS\$OUTPUT device and allows a command line user to respond to these requests interactively.

With 'Assist', the requests are sent to an operator terminal and mount commands are issued with assistance enabled (see MOUNT/ASSIST).

The default for an interactive process (which can be determined by using F\$MODE()) is 'NoAssist' and for any other process is 'Assist' (for example: a batch job).

13.1.34 New RMU/ALTER Feature to Modify the Area Header Root File Specification

The Oracle Rdb RMU/ALTER user could change the file specification of the storage area and snapshot files in the Rdb database root file using the DISPLAY FILE and DEPOSIT FILE commands. He could also change the root file specification in the database root file using the DEPOSIT ROOT SPECIFICATION and DISPLAY ROOT SPECIFICATION commands. However, the RMU/ALTER user previously could not change the database root file specification contained in the storage area file and snapshot file header block which identifies the database that the storage area or snapshot file belongs to. This enhancement adds this functionality.

The new syntax, which can only be used at the RMU/ALTER commands's "RdbALTER>" prompt, is the following, where "name" is the storage area name, and "id" is the storage area identification number in the database root file.

```
DISPLAY AREA_HEADER {name|id} [SNAPSHOT] SPECIFICATION
```

This command displays the current full root file specification in the storage area file or snapshot file header block for the storage area with the specified name or number.

```
DEPOSIT AREA_HEADER {name|id} [SNAPSHOT] SPECIFICATION  
[=DEV:[DIR]root_file_spec.rdb:version]
```

If the root file is not specified, this command deposits the current full root file specification in the database root in the storage area or snapshot file header block for the storage area with the specified name or number.

If the root file is specified, this command deposits the specified full database root file specification "DEV:[DIR]root_file.rdb;1" in the storage area or snapshot header block for the storage area with the specified name or number.

Any specified root file must exist or must have been changed by a previous RMU/ALTER DEPOSIT ROOT SPECIFICATION command. Only full VMS root file specifications are valid and must include a device, directory, extension and version number. Any changes to the area file headers will only be written to the actual area files when the "COMMIT" command is executed at the

"RdbALTER>" prompt. Any changes to area file headers since the last "COMMIT" command was issued can be undone by executing the "ROLLBACK" command at the "RdbALTER>" prompt. "COMMIT" and "ROLLBACK" are existing RMU/ALTER commands and affect any current uncommitted changes made in RMU/ALTER, not just changes to the storage area header files.

This new feature only allows modification of the root file specification in area headers, not other area header data. The "DISPLAY AREA_HEADER" command can be used with single file databases but the root file specification will always be blank, which is standard for single file databases. The "DEPOSIT AREA_HEADER" command cannot be used with single file databases since a root file specification is not specified in area headers in single file databases. To use either the DISPLAY or DEPOSIT AREA_HEADER command, the user must be attached to the database which the areas belong to, either by specifying the database name when issuing the RMU/ALTER command or by executing the "ATTACH" command from the "RdbALTER>" prompt.

The RMU/ALTER command should be used with caution and by those familiar with the internal structure of Oracle Rdb databases. All necessary interrelated changes need to be made to the database root file, the storage area and snapshot files, and the After Image Journaling files, etc, or the database will be corrupt. A full RMU/BACKUP of the database should be done previous to invoking RMU/ALTER and a full RMU/VERIFY of the database should be done once the RMU/ALTER changes have been committed and RMU/ALTER is exited.

The following example shows that RMU/ALTER is invoked specifying the multi-file database MULTIFILE_DB.RDB which has been previously backed up using RMU/BACKUP. The DEPOSIT ROOT SPECIFICATION command is used to change the full root file specification in the root file to DEVICE:[DIRECTORY]NEW_ROOT.RDB;1. Then the DEPOSIT AREA_HEADER command is used without a root file specification to change the root file specification in the storage area header to the current root file specification set by the previous DEPOSIT ROOT SPECIFICATION command for both the ST_AREA.RDA and ST_AREA.SNP files. Then the DEPOSIT AREA_HEADER command is used with a full root file specification "DEVICE:[DIRECTORY]NEW_ROOT.RDB;1" for both the ST_AREA.RDA and ST_AREA.SNP files. This just repeats the previous change but is included to show the use of the DEPOSIT AREA_HEADER COMMAND with and without an explicit root file specification. Then a COMMIT command is used to write the changes to the database files. After exiting RMU/ALTER, the name of the old root file is changed to DEVICE:[DIRECTORY]NEW_ROOT.RDB;1 from the VMS prompt. Then the RMU/VERIFY command is used to verify the integrity of the database. The DISPLAY AREA_HEADER command is used throughout to see the current root file specification in the storage area or snapshot file headers. In the display output "(marked)" means a change has been made but has not yet been committed. Note that, although this is not shown, in this case the area headers of all storage area and snapshot files in the database need to be changed to contain the new root file specification.

```
$ rmu/alter device:[directory]multifile_db
%RMU-I-ATTACH, now altering database "DEVICE:[DIRECTORY]MULTIFILE_DB.RDB;1"

RdbALTER> deposit root specification = DEVICE:[DIRECTORY]NEW_ROOT.RDB;1
          Root file specification is: "DEVICE:[DIRECTORY]NEW_ROOT.RDB;1"

RdbALTER> display root specification
(marked) Root file specification is: "DEVICE:[DIRECTORY]NEW_ROOT.RDB;1"

RdbALTER> display area_header st_area specification
Area ST_AREA:
          Root file specification is: "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1"
```

Oracle® Rdb for OpenVMS

```
RdbALTER> display area_header st_area snapshot specification
Area ST_AREA:
      Root file specification is: "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1"

RdbALTER> deposit area_header st_area specification
Area ST_AREA:
(marked) Root file specification is: "DEVICE:[DIRECTORY]NEW_ROOT.RDB;1"

RdbALTER> deposit area_header st_area snapshot specification
Area ST_AREA:
(marked) Root file specification is: "DEVICE:[DIRECTORY]NEW_ROOT.RDB;1"

RdbALTER> deposit area_header st_area specification =
      DEVICE:[DIRECTORY]NEW_ROOT.RDB;1
Area ST_AREA:
(marked) Root file specification is: "DEVICE:[DIRECTORY]NEW_ROOT.RDB;1"

RdbALTER> deposit area_header st_area snapshot specification =
      DEVICE:[DIRECTORY]NEW_ROOT.RDB;1
Area ST_AREA:
(marked) Root file specification is: "DEVICE:[DIRECTORY]NEW_ROOT.RDB;1"

RdbALTER> commit
RdbALTER> exit

$ RENAME DEVICE:[DIRECTORY]MULTIFILE_DB.RDB;1 DEVICE:[DIRECTORY]NEW_ROOT.RDB;1
$ RMU/VERIFY/ALL DEVICE:[DIRECTORY]NEW_ROOT.RDB
```

13.1.35 Create Index Supports the REVERSE Keyword to Create Reverse Key Indices

Bug 5710904

This release of Oracle Rdb introduces a new sorted index attribute: REVERSE. A REVERSE key index reverses the bits of the key value before entering it in the index. Conceptually, a key value 24538 would become 83542 in the index (in reality, the bits of the key are reversed as opposed to the digits shown here). Reversing the key value can be particularly useful for indexing data, such as sequence numbers, where each new key value is greater than the prior value. This may help distribute access within the index among the leaf nodes rather than concentrating the access on the lower right corner of the index.

Reverse key indexes may be helpful in several situations including:

- ◆ High volume transaction processing systems where they can help reduce contention for index nodes.
- ◆ Applications that delete data that is older on average (with lower values of the sequence) before deleting newer data because in traditional B-trees, many index nodes may end up containing few values, with a commensurate increase in unused space.

A reverse key index can be used for direct key lookup in a similar fashion to hash indexes. Range scans, partial key lookups and certain index optimizations are not applicable to reverse key indexes. See the SQL Reference Manual for further details.

13.1.36 Support for New Syntax for Sequence Generator Statements

This release of Oracle Rdb enhances the support for CREATE SEQUENCE, ALTER SEQUENCE and the IDENTITY clause by adding features from the ANSI and ISO SQL Language Standard.

- ◆ Alternate keywords supported in the ALTER SEQUENCE and CREATE SEQUENCE statements

The original Rdb implementation used single keywords for negated items: NOMAXVALUE, NOMINVALUE, NOCYCLE, NOORDER, NORANDOM, and NOWAIT. However, in the SQL Standard that was published after Oracle Rdb was released, SQL uses a separate NO keyword. Rdb now supports both formats.

The following are equivalent clauses: NOMAXVALUE and NO MAXVALUE, NOMINVALUE and NO MINVALUE, NOCYCLE and NO CYCLE, NOORDER and NO ORDER, NORANDOM and NO RANDOM, NOWAIT and NO WAIT.
- ◆ Support for IDENTITY column creation

IDENTITY can now be followed by a list of sequence attributes: START WITH, INCREMENT BY, MAXVALUE, NO MAXVALUE, MINVALUE, NO MINVALUE, CYCLE, NO CYCLE, ORDER, and NO ORDER.

The previous numeric list that represented a starting with and an optional increment value is supported for backward compatibility.

Any IDENTITY created in Oracle Rdb Release 7.3 or later will default to a NO CYCLE sequence, unlike the default in prior releases. If CYCLE is desired, then include the CYCLE clause as part of the IDENTITY specification.

```
SQL> create table SAMPLE1
cont>      (a integer identity (cycle minvalue 100 no maxvalue cache 2000)
cont>      ,b integer
cont>      );
SQL>
```

- ◆ Support for IDENTITY clause source

SQL now captures the source for the IDENTITY clause and therefore SHOW TABLE includes more details than previous versions.

```
SQL> show table (column) SAMPLE1;
Information for table SAMPLE1

Columns for table SAMPLE1:
Column Name          Data Type          Domain
-----
A                    INTEGER
  Computed:          Identity (cycle minvalue 100 no maxvalue cache 2000)
B                    INTEGER

SQL>
```

As in prior versions, you can use the SHOW SEQUENCE command with the name of the table to show details of the identity sequence.

```
SQL> show sequence SAMPLE1;
SAMPLE1
Sequence Id: 4
```

```
An identity column sequence.
Initial Value: 100
Minimum Value: 100
Maximum Value: (none)
Next Sequence Value: 100
Increment by: 1
Cache Size: 2000
No Order
Cycle
No Randomize
Wait
Comment:          column IDENTITY sequence
SQL>
```

13.1.37 RMU/SET AUDIT Supports MODULE, ROUTINE, SEQUENCE and VIEW

This release of Oracle Rdb adds several enhancements to the RMU/SET AUDIT command. The /ENABLE=DACCESS and /DISABLE=DACCESS qualifiers now accept the following new keywords for auditing.

- ◆ SEQUENCE – The SEQUENCE keyword specifies the names of sequences, either explicitly, as in this command:

```
$ RMU/SET AUDIT/ENABLE=DACCESS=SEQUENCE:NEW_DEPT MF_PERSONNEL
```

or via a wildcard as in this command:

```
$ RMU/SET AUDIT/ENABLE=DACCESS=SEQUENCE:*ID*
```

Only user defined sequences will be selected by this type of wildcard command. Those created by Oracle Rdb must be completely specified.

- ◆ ROUTINE – The ROUTINE keyword specifies the names of functions and procedures, either explicitly, as in this command:

```
$ RMU/SET AUDIT/ENABLE=DACCESS=ROUTINE:CHECKSUM13 ACCOUNTING
```

or via a wildcard as in this command:

```
$ RMU/SET AUDIT/ENABLE=DACCESS=ROUTINE:CHECKSUM%%
```

Only user defined routines will be selected by this type of wildcard command. Those created by Oracle Rdb must be completely specified.

- ◆ MODULE – The MODULE keyword specifies the names of modules, either explicitly, as in this command:

```
$ RMU/SET AUDIT/ENABLE=DACCESS=MODULE:PROD_SUPPORT MF_PERSONNEL
```

or via a wildcard as in this command:

```
$ RMU/SET AUDIT/ENABLE=DACCESS=MODULE:UTIL*
```

Only user defined modules will be selected by this type of wildcard command. Those created by Oracle Rdb must be completely specified. Note also that routines defined with the clause `USAGE IS LOCAL` will not be selected by wildcards. Such routines can only be activated by routines in the same module.

The `MODULE` keyword provides a time saving shortcut for auditing related routines. In a similar way that routine access control is inherited from the containing module, audit and alarm settings are inherited from the owning module when a routine is first referenced. When a subsequent `ALTER MODULE ... ADD FUNCTION`, or `ALTER MODULE ... ADD PROCEDURE` statement is used, these new routines will also inherit audit and alarm settings from the module.

- ◆ **VIEW** – The `VIEW` keyword specifies the names of views, either explicitly, as in this command:

```
$ RMU/SET AUDIT/ENABLE=DACCESS=VIEW:CURRENT_JOB MF_PERSONNEL
```

or via a wildcard as in this command:

```
$ RMU/SET AUDIT/ENABLE=DACCESS=VIEW:CURRENT*
```

Only user defined views will be selected by this type of wildcard command. Views created by Oracle Rdb must be completely specified.

In prior releases, views could be marked for audit using the `TABLE` keyword. The `TABLE` keyword remains a superset of `VIEW` and selects both table and view objects. However, to perform wildcard selection of views you must use the `VIEW` keyword.

Note

At this time `RMU/SET AUDIT` accesses multischema databases using `MULTISHEMA IS OFF`. Therefore, the external (possibly generated) names must be specified for the `/ENABLE=DACCESS` and `/DISABLE=DACCESS` qualifiers.

13.1.38 RMU/SHOW AUDIT Supports MODULE, ROUTINE, SEQUENCE and VIEW

This release of Oracle Rdb adds several enhancements to the `RMU/SHOW AUDIT` command.

The `/DACCESS` qualifier now accepts the following new keywords for auditing.

- ◆ **SEQUENCE**

The `SEQUENCE` keyword reports those sequences that have audit and/or alarm settings.

- ◆ ROUTINE
The ROUTINE keyword reports those functions and procedures that have audit and/or alarm settings. Note that routines defined within a module may inherit audit and alarm settings from the containing module and will not be reported in such cases.
- ◆ MODULE
The MODULE keyword reports those modules that have audit and/or alarm settings.
- ◆ VIEW
The VIEW keyword reports those views that have audit and/or alarm settings. Note that the TABLE keyword reports both tables and views that have auditing enabled.

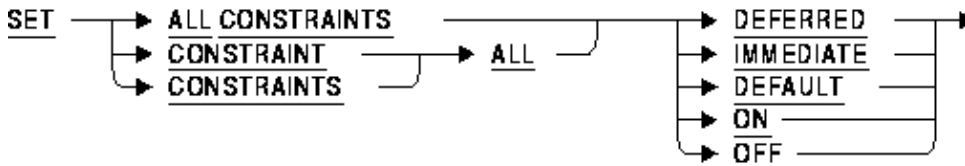
In addition to these changes, the /DACCESS=TABLE option shows that the audited relation is a view or a table.

Note

At this time, RMU/SHOW AUDIT accesses multischema databases using MULTISCHEMA IS OFF. Therefore, the external (possibly generated) names will be displayed for all objects.

13.1.39 SQL Now Supports SQL Standard Syntax for SET CONSTRAINTS ALL Statement

This release of Oracle Rdb now supports the ANSI/ISO SQL Standard statement SET CONSTRAINTS in addition to the older Rdb syntax.



The existing SET ALL CONSTRAINTS statement is retained for backward compatibility.

Please see the Oracle SQL/Services Release 7.3.1.1 Release Notes, Note 4.3.1 SET CONSTRAINTS Command Now Translated to Oracle Rdb Format, for more information.

This syntax has been implemented in Oracle Rdb Release 7.3.1.0.

13.1.40 Support ANSI and ISO SQL Standard Length Units

This release of Oracle Rdb allows the specification of the length used by data type definitions and string handling functions. In prior releases, the SET CHARACTER LENGTH statement had to precede the CREATE, DECLARE, or ALTER data definition (DDL) statements, or any usage of the SUBSTRING function in a data manipulation (DML) statement to effect the choice of character or octet units for string length and position values.

Oracle® Rdb for OpenVMS

The following functions and data types are affected by this change.

- ◆ The SUBSTRING function now includes an optional USING clause to specify that either OCTETS or CHARACTERS units are used for the FROM and FOR clauses.

```
SUBSTRING ( char-value-expr
            FROM start-position
            [ FOR string-length ]
            [ USING { CHARACTERS | OCTETS } ] )
```

- ◆ The new OVERLAY function includes an optional USING clause to specify that either OCTETS or CHARACTERS units are used for the FROM and FOR clauses.

```
OVERLAY ( char-value-expr
          PLACING char-value-expr
          FROM start-position
          [ FOR string-length ]
          [ USING { CHARACTERS | OCTETS } ] )
```

- ◆ CHARACTER, NATIONAL CHARACTER (NCHAR), CHARACTER VARYING (VARCHAR), NATIONAL CHARACTER VARYING (NCHAR VARYING) and related types now accept an optional OCTETS or CHARACTERS option, as in the following example.

```
CHAR (20 CHARACTERS)
NATIONAL CHARACTER VARYING (300 OCTETS)
```

- ◆ The SIZE IS clause of the CREATE INDEX statement also accepts an optional OCTETS or CHARACTERS option.

```
SIZE IS <n> [ CHARACTERS | OCTETS ]
```

Using these clauses will override any current setting of the SET CHARACTER LENGTH statement or the SET DIALECT statement.

The following shows examples of these changes.

```
SQL> create database
cont>     filename SAMPLE_DB
cont>     national character set DEC_KANJI
cont> ;
SQL>
SQL> set character length 'characters';
SQL>
SQL> create table EMPLOYEES
cont>     (name          nchar(10 characters)
cont>     ,department   char(10 octets)
cont>     ,comments      character varying (300 characters)
cont>     );
SQL>
SQL> create index EMPLOYEES_COMMENTS
cont>     on EMPLOYEES (comments size is 30 characters)
cont> ;
SQL>
SQL> declare :fl char(20 octets);
SQL> select rdb$flags into :fl from rdb$database;
SQL> print :fl;
FL
131328
SQL>
SQL> select name
cont>     ,department
```



```

cont>      ,substring (comments from 1 for 30 using characters) as part_comment
cont> from EMPLOYEES
cont> where comments starting with 'Review: ';
0 rows selected
SQL>
SQL> commit;
SQL>

```

13.1.41 New SET FLAGS Clause Supported by CREATE and ALTER PROFILE

In this release of Oracle Rdb, the CREATE and ALTER PROFILE statements have been enhanced with a SET FLAGS clause. This new clause is related to the SET FLAGS statement; refer to that documentation for the list of available keywords that can be specified.

The string associated with the SET FLAGS clause is saved with the created profile. Any user that has this assigned profile will implicitly execute SET FLAGS during session start.

Note

Please notice that some SET FLAGS keywords affect actions during database attach and so have no action when defined within a profile. For example, DATABASE_PARAMETERS, generates minimal effects in such cases.

The following example shows the creation of a profile with flags and the assigning of the profile to a user.

```

SQL> create profile SF_USER
cont> set flags
cont> 'old_cost_model,noindex_column_group,optimization_level(total_time) '
cont> ;
SQL>
SQL> alter user SMITH
cont> profile SF_USER;
SQL>

```

When this user (SMITH) attaches to a database (ATTACH, CONNECT, DECLARE ALIAS), or uses SET SESSION AUTHORIZATION, a SET FLAGS statement will implicitly be executed using this string of keywords.

Note that the CREATE PROFILE and ALTER PROFILE statements will validate the listed keywords.

```

SQL> alter profile SF_USER
cont> set flags 'THIS_OLD_HOUSE'
cont> ;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-EXT_ERR, Oracle Rdb extension error
-RDMS-E-UNKAMBFLAG, 'THIS_OLD_HOUSE' is an unknown or ambiguous flag name
SQL>

```

The clause NO SET FLAGS can be used to remove any flags associated with the profile. All users

assigned that profile will no longer perform a SET FLAGS action during session start.

```
SQL> show profile SF_USER
      SF_USER
      Flags: "old_cost_model,noindex_column_group,optimization_level(total_time)"
SQL> alter profile SF_USER
cont> no set flags;
SQL> show profile SF_USER
      SF_USER
SQL>
```

13.1.42 New Support for SAVEPOINT Syntax and Semantics

This release of Oracle Rdb adds support for a SAVEPOINT. The SAVEPOINT feature allows the programmer to place a marker within a transaction that can later be used to undo part of the transaction using the ROLLBACK TO SAVEPOINT statement. Additionally, this marker can be freed using the RELEASE SAVEPOINT statement.

Note

At this time, Oracle Rdb only supports a single active SAVEPOINT per transaction.

Some SQL DDL statements currently use this feature to implement SQL semantics and therefore mixing SAVEPOINT and these statements is not supported. Some SQL statements that use SAVEPOINT include: GRANT and REVOKE using * wildcard object names, SET CONSTRAINT MODE 'ON', some forms of ALTER MODULE statement, and an INSERT ... SELECT statement that uses two database aliases.

13.1.42.1 SAVEPOINT Statement

The SAVEPOINT statement establishes a marker in the current transaction that allows the programmer to undo part of the transaction (using ROLLBACK TO SAVEPOINT) without resorting to a full transaction ROLLBACK.

Syntax

savepoint-statement =

→ **SAVEPOINT** → **savepoint-name** →
 └─→ **alias-name .** ─┘

Arguments

- ◆ **alias-name**
 This optional alias name can be used to target a specific database alias. If no alias-name is provided, then the current default database will be used.

- ◆ `savepoint-name`
Name of a unique identifier for this savepoint. This name will be used with subsequent `ROLLBACK TO SAVEPOINT` and `RELEASE SAVEPOINT` statements.

Usage Notes

- ◆ If the `SAVEPOINT` statement is used more than once with the same name, then the prior `SAVEPOINT` is destroyed and replaced with this new location.
- ◆ Any established savepoints will be discarded by a `ROLLBACK` statement (which does not use the `TO SAVEPOINT` clause), and by a `COMMIT` statement.
- ◆ If more savepoints are created than are supported by Rdb, then the error `RDB$_EXCESS_SVPT` will be raised. `SQLCODE` will be returned as `-880` and `SQLSTATE` will be returned as `3B002`.

```
%RDB-E-EXCESS_SVPT, maximum number of savepoints are already active -
"BOOK2" failed
```

- ◆ The `SAVEPOINT` statement may not be used in a SQL function definition nor can it be called indirectly from a function.
- ◆ The `SAVEPOINT` statement may not be called indirectly from a trigger action.
- ◆ A `SAVEPOINT` statement is only valid if a transaction is in progress. This can be either a `READ WRITE` or `READ ONLY` transaction. Note that temporary tables can be updated during a read only transaction.

```
SQL> commit;
SQL> savepoint BK;
%RDB-E-NOTXNINPRGS, no transaction is in progress
-RDB-E-SVPT_NOALLOWED, a savepoint may not be established in this context -
"BK" failed
```

The following example shows the use of the `SAVEPOINT` statement. Note that reusing the savepoint name will re-establish that marker and so affect different rows in the transaction.

```
SQL> declare local temporary table module.SAMPLE
cont>      (a integer)
cont>      on commit preserve rows
cont> ;
SQL>
SQL> --
SQL> set transaction read only;
SQL>
SQL> insert into module.SAMPLE values (1);
1 row inserted
SQL>
SQL> -- Establish the initial marker
SQL> savepoint BOOK_IT;
SQL>
SQL> insert into module.SAMPLE values (2);
1 row inserted
SQL> insert into module.SAMPLE values (3);
1 row inserted
SQL>
SQL> table module.SAMPLE;
      A
      1
      2
      3
3 rows selected
SQL>
```

```

SQL> -- Move the marker
SQL> savepoint BOOK_IT;
SQL>
SQL> insert into module.SAMPLE values (4);
1 row inserted
SQL>
SQL> rollback to savepoint BOOK_IT;
SQL>
SQL> table module.SAMPLE;
      A
      1
      2
      3
3 rows selected
SQL>
SQL> commit;
SQL>

```

13.1.42.2 RELEASE SAVEPOINT Statement

The RELEASE SAVEPOINT Statement destroys the named savepoint established by the SAVEPOINT statement. Changes made by the transaction are unaffected by this statement.

Syntax

release-savepoint-statement =



Arguments

- ◆ **alias-name**
This optional alias name can be used to target a specific database alias. If no alias-name is provided, then the current default database will be used.
- ◆ **savepoint-name**
Name of a unique identifier for this savepoint. This name is declared using the SAVEPOINT statement.

Usage Notes

- ◆ If no established savepoint exists with this name, then the error RDB\$_BAD_SVPT_HANDLE will be raised. SQLCODE will be returned as -882 and SQLSTATE will be returned as 3B001.

%RDB-E-BAD_SVPT_HANDLE, invalid savepoint handle - "BOOKMARK2" is unknown
- ◆ The RELEASE SAVEPOINT statement may not be used in a SQL function definition nor can it be called indirectly from a function.
- ◆ The RELEASE SAVEPOINT statement may not be called indirectly from a trigger action.

The following example shows the use of the RELEASE SAVEPOINT statement.

```
SQL> set transaction read write;
```

```

SQL>
SQL> insert into module.SAMPLE values (1);
1 row inserted
SQL>
SQL> savepoint BOOK_IT;
SQL>
SQL> insert into module.SAMPLE values (2);
1 row inserted
SQL> insert into module.SAMPLE values (3);
1 row inserted
SQL>
SQL> table module.SAMPLE;
      A
      1
      2
      3
3 rows selected
SQL>
SQL> release savepoint BOOK_IT;
SQL>
SQL> insert into module.SAMPLE values (4);
1 row inserted
SQL>
SQL> table module.SAMPLE;
      A
      1
      2
      3
      4
4 rows selected
SQL>
SQL> commit;
SQL>

```

13.1.42.3 ROLLBACK TO SAVEPOINT Statement

The ROLLBACK TO SAVEPOINT statement destroys the named savepoint established by the SAVEPOINT statement and removes all database changes made from the time the SAVEPOINT statement established the named savepoint.

Syntax

rollback-savepoint-statement =



Arguments

- ◆ alias-name
This optional alias name can be used to target a specific database alias. If no alias-name is provided then the current default database will be used.
- ◆ savepoint-name

Name of a unique identifier for this savepoint. This name is declared using the SAVEPOINT statement.

Usage Notes

- ◆ If no established savepoint exists with this name then the error RDB\$_BAD_SVPT_HANDLE will be raised. SQLCODE will be returned as -882 and SQLSTATE will be returned as 3B001.
- %RDB-E-BAD_SVPT_HANDLE, invalid savepoint handle - "BOOKMARK2" is unknown
- ◆ The ROLLBACK TO SAVEPOINT statement may not be used in a SQL function definition nor can it be called indirectly from a function.
 - ◆ The ROLLBACK TO SAVEPOINT statement may not be called indirectly from a trigger action.

The following example shows the use of SAVEPOINT and ROLLBACK TO SAVEPOINT to exclude rows inserted during the transaction. In an actual application, the ROLLBACK TO SAVEPOINT statement would probably be within a conditional statement such as IF-THEN-ELSE or CASE statement.

```
SQL> declare local temporary table module.SAMPLE
cont>     (a integer)
cont>     on commit preserve rows
cont> ;
SQL>
SQL> set transaction read only;
SQL>
SQL> insert into module.SAMPLE values (1);
1 row inserted
SQL>
SQL> savepoint BOOK_IT;
SQL>
SQL> insert into module.SAMPLE values (2);
1 row inserted
SQL> insert into module.SAMPLE values (3);
1 row inserted
SQL>
SQL> table module.SAMPLE;
      A
      1
      2
      3
3 rows selected
SQL>
SQL> rollback to savepoint BOOK_IT;
SQL>
SQL> insert into module.SAMPLE values (4);
1 row inserted
SQL>
SQL> table module.SAMPLE;
      A
      1
      4
2 rows selected
SQL>
SQL> commit;
SQL>
```

13.1.43 New OPTIMIZE OUTLINE Clause Allows Outline Specification

Bug 2835544

This release of Oracle Rdb extends the use of query outlines so they can be specified in-line with SELECT, DELETE, UPDATE, INSERT and BEGIN PRAGMA statements.

In some cases, using the CREATE OUTLINE statement to define a query outline might not be possible or permitted. The OUTLINE option is part of the OPTIMIZE clause and allows the query outline to be packaged with the query.

The following example shows a query modified with an outline.

```
SQL> set flags 'strategy,detail(2),request_name';
SQL> select last_name, middle_initial, first_name
cont> from employees2
cont> where last_name = 'Toliver' and first_name = 'Alvin'
cont> optimize
cont>     as test3
cont>     outline (
cont>         mode 0
cont>         as (
cont>             query (
cont>                 subquery (
cont>                     EMPLOYEES2 0  access path index  E3_INDEX
cont>                 )
cont>             )
cont>         )
cont>         compliance optional
cont>         execution options (total time)
cont>     );
~Query Name: "TEST3"
~S: Outline "(unnamed)" used
Tables:
  0 = EMPLOYEES2
Leaf#01 BgrOnly 0:EMPLOYEES2 Card=100
  Bool: (0.LAST_NAME = 'Toliver') AND (0.FIRST_NAME = 'Alvin')
  BgrNdx1 E3_INDEX [1:1] Fan=14
  Keys: 0.LAST_NAME = 'Toliver'
  LAST_NAME      MIDDLE_INITIAL  FIRST_NAME
  Toliver        A.                Alvin
1 row selected
SQL>
```

This example was produced by using the output from the SET FLAGS 'OUTLINE' statement and capturing the portion that defines the outline actions.

Usage Notes

- ◆ If MODE is specified with the OPTIMIZE OUTLINE clause, then the specified query outline will be ignored unless that mode is established using the SET FLAGS 'MODE(n)' statement or if the logical name RDMS\$BIND_OUTLINE_MODE is used to define a matching mode value.

```

set flags 'strategy';

call DELETE_EMP ('Toliver', 'Alvin');
~S: Specified mode (99) does not match current mode - outline ignored
~Query Name: "TEST6"
Tables:
  0 = EMPLOYEES2
Conjunct: (0.LAST_NAME = <var0>) AND (0.FIRST_NAME = <var1>)
Get      Temporary relation      Retrieval by index of relation 0:EMPLOYEES2
Index name  E1_INDEX [2:2]
Keys: (0.LAST_NAME = <var0>) AND (0.FIRST_NAME = <var1>)

set flags 'mode(99)';

call DELETE_EMP ('Toliver', 'Alvin');
~Query Name: "TEST6"
~S: Outline "(unnamed)" used
Tables:
  0 = EMPLOYEES2
Conjunct: (0.LAST_NAME = <var0>) AND (0.FIRST_NAME = <var1>)
Get      Retrieval sequentially of relation 0:EMPLOYEES2

```

- ◆ The SET FLAGS 'OUTLINE' statement can be used to have the Rdb optimizer provide a template query outline that can then be modified and incorporated into the problem query.
- ◆ Refer to the CREATE OUTLINE statement for detail of the query outline definition language.

13.1.44 RMU/DUMP/HEADER=ROW_CACHE Now Displays Whether Row Cache is Enabled

Prior to Oracle Rdb Release 7.3.1.0, the RMU RMU/DUMP/HEADER=ROW_CACHE command displayed the database-wide Row Cache definitions for an Rdb database and the RMU/DUMP/HEADER/AREA command displayed the Row Cache to be used and whether the Row Cache feature was enabled for a particular storage area. The RMU/DUMP/HEADER=ROW_CACHE command will now display whether Row Cache is enabled for ANY database storage areas. The RMU/DUMP/HEADER=ROW_CACHE command will now display the following:

```
Row caching is enabled
```

if the Row Cache feature is currently enabled for one or more database storage areas. If the Row Cache feature is not currently enabled for at least one database storage area the RMU/DUMP/HEADER=ROW_CACHE command will now display the following:

```
Row caching is disabled
```

Note that Row Cache definitions can exist in the database but a Row Cache must be enabled for a particular database storage area. The RMU/DUMP/HEADER/AREA command must still be used to see the per area Row Cache settings and whether Row Cache is currently enabled or disabled for a particular storage area.

In the following example, a database is defined with two Row Caches which are assigned to storage areas for which row caching is enabled. An RMU/DUMP/HEADER=AREA command shows the row cache parameters and whether or not row caching is enabled for each storage area. The RMU/DUMP/HEADER=ROW_CACHE command shows the database-wide Row Cache definitions. The new display "Row caching is enabled" is output since row caching is enabled for at least one

Oracle® Rdb for OpenVMS

storage area (in this case for all storage areas).

```
$ sql
create database filename DEVICE:[DIRECTORY]:ROW_CACHEDB
  number of cluster nodes is 1
  reserve 5 cache slots
  reserve 5 storage areas
  row cache is enabled (checkpoint updated rows to database)
  create cache tbl_phys_cache row length is 44 bytes cache size is 40 rows
  create cache idx_phys_cache row length is 432 bytes cache size is 10 rows
  create storage area tbl_sto_ar_low filename rcachedb_emp_sal_low
    cache using tbl_phys_cache
  create storage area tbl_sto_ar_high filename rcachedb_emp_sal_high
    cache using tbl_phys_cache
  create storage area idx_sto_ar filename rcachedb_emp_no
    cache using idx_phys_cache;

exit;
$ RMU/DUMP/HEADER=AREA DEVICE:[DIRECTORY]ROW_CACHEDB
*-----
* Oracle Rdb V7.3-01                               dd-mmm-yyyy hh:mm:ss.xxxx
*
* Dump of Database header
*   Database: DEVICE:[DIRECTORY]ROW_CACHEDB.RDB;1
*
*-----

Database Parameters:
  Root filename is "DEVICE:[DIRECTORY]ROW_CACHEDB.RDB;1"

Storage area "RDB$SYSTEM"

  Row Caching...
    - Row caching is enabled
    - No row cache is defined for this area

Storage area "TBL_STO_AR_LOW"

  Row Caching...
    - Row caching is enabled
    - Row cache ID is 1

Storage area "TBL_STO_AR_HIGH"

  Row Caching...
    - Row caching is enabled
    - Row cache ID is 1

Storage area "IDX_STO_AR"

  Row Caching...
    - Row caching is enabled
    - Row cache ID is 2

$ RMU/DUMP/HEADER=ROW_CACHE DEVICE:[DIRECTORY]:ROW_CACHEDB
*-----
* Oracle Rdb V7.3-01                               dd-mmm-yyyy hh:mm:ss.xxxx
*
* Dump of Database header
*   Database: DEVICE:[DIRECTORY]ROW_CACHEDB.RDB;1
*
*-----
```

Oracle® Rdb for OpenVMS

Database Parameters:

Root filename is "DEVICE:[DIRECTORY]ROW_CACHEDB.RDB;1"

Row Caches...

- Active row cache count is 2
- Reserved row cache count is 5
- Checkpoint information
 - No time interval is specified
 - Default source is updated rows
 - Default target is database
 - Default backing file directory is database directory
 - RUJ Global Buffers are enabled
 - No RCS sweep time interval is specified
- WARNING: After-image journaling is disabled
- WARNING: Fast commit is disabled

Row caching is enabled

Row cache "TBL_PHYS_CACHE"

Cache ID number is 1

Allocation...

- Row slot count is 40
 - Snapshot slot count is 1000
 - Snapshots in cache disabled
- Maximum row size allowed in cache is 44 bytes
- Working set count is 10
- Maximum slot reservation count is 20
- Row replacement is enabled

Sweeping...

- Sweep row count is 0
- Maximum batch I/O count is 3000

Checkpointing...

- Source is updated rows (database default)
- Target is database (database default)
- No checkpoint information available
- Checkpoint sequence is 0

Files...

- Derived cache file directory is "DEVICE:[DIRECTORY]"
- File allocation is 100 blocks
- File extension is 100 blocks

Hashing...

- Hash value for logical area DBIDs is 31
- Hash value for page numbers is 7

Shared Memory...

- Global Section Name is "RDM73R\$1\$DGA22084690010000000000001"
- Shared memory section requirement is 16,384 bytes (1MB)

Row cache "IDX_PHYS_CACHE"

Cache ID number is 2

Allocation...

- Row slot count is 10
 - Snapshot slot count is 1000
 - Snapshots in cache disabled
- Maximum row size allowed in cache is 432 bytes
- Working set count is 10
- Maximum slot reservation count is 20
- Row replacement is enabled

Sweeping...

- Sweep row count is 0
- Maximum batch I/O count is 3000

Checkpointing...

- Source is updated rows (database default)
- Target is database (database default)

Oracle® Rdb for OpenVMS

```
- No checkpoint information available
- Checkpoint sequence is 0
Files...
- Derived cache file directory is "DEVICE:[DIRECTORY]"
- File allocation is 100 blocks
- File extension is 100 blocks
Hashing...
- Hash value for logical area DBIDs is 31
- Hash value for page numbers is 7
Shared Memory...
- Global Section Name is "RDM73R$1$DGA22084690010000000000002"
- Shared memory section requirement is 16,384 bytes (1MB)
```

Now an SQL statement is used to disable row caching for all storage areas. The RMU/DUMP/HEADER=AREA display shows that row caching is disabled for all storage areas and the RMU/DUMP/HEADER=ROW_CACHE command also displays "Row caching is disabled" since row caching is now disabled for all storage areas.

```
$ SQL
alter database filename DEVICE:[DIRECTORY]ROW_CACHEDB
row cache is disabled;
exit;
$ RMU/DUMP/HEADER=AREA DEVICE:[DIRECTORY]:ROW_CACHEDB
*-----*
* Oracle Rdb V7.3-01                               dd-mmm-yyyy hh:mm:ss.xxxx
*
* Dump of Database header
*   Database: DEVICE:[DIRECTORY]ROW_CACHEDB.RDB;1
*
*-----*

Database Parameters:
  Root filename is "DEVICE:[DIRECTORY]ROW_CACHEDB.RDB;1"

Storage area "RDB$SYSTEM"

  Row Caching...
    - Row caching is disabled
    - No row cache is defined for this area

Storage area "TBL_STO_AR_LOW"

  Row Caching...
    - Row caching is disabled
    - Row cache ID is 1

Storage area "TBL_STO_AR_HIGH"
  Row Caching...
    - Row caching is disabled
    - Row cache ID is 1

Storage area "IDX_STO_AR"
  Row Caching...
    - Row caching is disabled
    - Row cache ID is 2

$ RMU/DUMP/HEADER=ROW_CACHE DEVICE:[DIRECTORY]ROW_CACHEDB
*-----*
* Oracle Rdb v7.3-01                               dd-mmm-yyyy hh:mm:ss.xxxx
*
```

Oracle® Rdb for OpenVMS

```
* Dump of Database header
*   Database: DEVICE:[DIRECTORY]ROW_CACHEDB.RDB;1
*
*-----
```

Database Parameters:

```
Root filename is "DEVICE:[DIRECTORY]ROW_CACHEDB.RDB;1"
Row Caches...
- Active row cache count is 2
- Reserved row cache count is 5
- Checkpoint information
  No time interval is specified
  Default source is updated rows
  Default target is database
  Default backing file directory is database directory
  RUJ Global Buffers are enabled
  No RCS sweep time interval is specified
- WARNING: After-image journaling is disabled
- WARNING: Fast commit is disabled
```

Row caching is disabled

Row cache "TBL_PHYS_CACHE"

```
Cache ID number is 1
Allocation...
- Row slot count is 40
  - Snapshot slot count is 1000
  - Snapshots in cache disabled
- Maximum row size allowed in cache is 44 bytes
- Working set count is 10
- Maximum slot reservation count is 20
- Row replacement is enabled
Sweeping...
- Sweep row count is 0
- Maximum batch I/O count is 3000
Checkpointing...
- Source is updated rows (database default)
- Target is database (database default)
- No checkpoint information available
- Checkpoint sequence is 0
Files...
- Derived cache file directory is "DEVICE:[DIRECTORY]"
- File allocation is 100 blocks
- File extension is 100 blocks
Hashing...
- Hash value for logical area DBIDs is 31
- Hash value for page numbers is 7
Shared Memory...
- Global Section Name is "RDM73R$1$DGA22084690010000000000001"
- Shared memory section requirement is 16,384 bytes (1MB)
```

Row cache "IDX_PHYS_CACHE"

```
Cache ID number is 2
Allocation...
- Row slot count is 10
  - Snapshot slot count is 1000
  - Snapshots in cache disabled
- Maximum row size allowed in cache is 432 bytes
- Working set count is 10
- Maximum slot reservation count is 20
- Row replacement is enabled
Sweeping...
```

Oracle® Rdb for OpenVMS

```
- Sweep row count is 0
- Maximum batch I/O count is 3000
Checkpointing...
- Source is updated rows (database default)
- Target is database (database default)
- No checkpoint information available
- Checkpoint sequence is 0
Files...
- Derived cache file directory is "DEVICE:[DIRECTORY]"
- File allocation is 100 blocks
- File extension is 100 blocks
Hashing...
- Hash value for logical area DBIDs is 31
- Hash value for page numbers is 7
Shared Memory...
- Global Section Name is "RDM73R$1$DGA22084690010000000000002"
- Shared memory section requirement is 16,384 bytes (1MB)
$ EXIT
```

13.1.45 RMU/LOAD Now Supports CSV Formatted Files

This release of Oracle Rdb adds limited support for the CSV (comma separated list of values) format used by many tools to load data. RMU/LOAD now supports the keyword CSV, which is a variation of the DELIMITED_TEXT format currently supported by Oracle Rdb.

Usage Notes

FORMAT=CSV support is almost identical to FORMAT=DELIMITED_TEXT with some additional semantics:

- ◆ RMU/UNLOAD will create TIMESTAMP(2) format strings that are compatible with various CSV knowledgeable tools (such as Microsoft EXCEL). RMU/LOAD will implicitly convert these strings to DATE VMS during load.
- ◆ The first row is a list of column names. RMU/LOAD will implicitly skip this first row. If the CSV file is generated with multiple header lines, use the /SKIP qualifier to skip the additional lines.
- ◆ The file type defaults to .CSV

13.1.46 RMU/UNLOAD Now Supports CSV Formatted Files

This release of Oracle Rdb adds support for the CSV (comma separated list of values) format used by many tools to load data. RMU/UNLOAD now supports the keyword CSV which is a variation of the DELIMITED_TEXT format currently supported by Oracle Rdb.

Usage Notes

- ◆ Implicit conversion of DATE VMS to TIMESTAMP(2) so that formatting of text string is compatible with various CSV knowledgeable tools (such as Microsoft EXCEL).
- ◆ The first row is a list of column names. These values are formatted using the same PREFIX, SUFFIX, SEPARATOR and TERMINATOR strings as defined for the table data.

- ◆ The list of column names is re-generated when the unload file is reopened. See REOPEN_COUNT qualifier.
- ◆ The file type defaults to .CSV.

Examples

The following example shows using the RMU/UNLOAD command to generate a portable data file. The output RRD (record definition) file is suppressed using the NOFILE keyword as it is usually not useful for the target tool. The TRIM keyword is used to remove unnecessary padding spaces.

Example 13–1 Using CSV format for Microsoft EXCEL export

```
$ rmu/unload-
  /record=(nofile,format=csv,trim=trailing,term=";") -
  sql$database -
  work_status -
  ws.csv
%RMU-I-DATRECUNL,    4 data records unloaded.
$ ty ws.csv
"STATUS_CODE", "STATUS_NAME", "STATUS_TYPE";
"0", "INACTIVE", "RECORD EXPIRED";
"1", "ACTIVE", "FULL TIME";
"2", "ACTIVE", "PART TIME";
```

This example changes the delimiters for the data as required by the target loading tool.

Example 13–2 Using options to change delimiters in a CSV formatted file

```
$ rmu/unload-
  /record=(nofile,format=csv,trim=trailing,-
           term=";",pref="{",suff="}") -
  sql$database -
  current_job -
  cj.csv
...
{LAST_NAME}, {FIRST_NAME}, {EMPLOYEE_ID}, {JOB_CODE}, {DEPARTMENT_CODE},
{SUPERVISOR_ID}, {JOB_START};
{Toliver}, {Alvin}, {00164}, {DMGR}, {MBMN}, {00228}, {1981-09-21 00:00:00.00};
...
```

13.1.47 RMU/UNLOAD Supports BITMAPPED_SCAN Optimize Option

This release of Oracle Rdb adds the keyword BITMAPPED_SCAN to the RMU/UNLOAD/Optimize qualifier.

- ◆ Bitmapped_scan
 - This option requests that the Rdb optimizer attempt to perform bitmapped scan when accessing multiple indices during the unload. This option is particularly useful for RMU/UNLOAD from complex views.
 - This option cannot be specified at the same time as the Sequential_Access option.
 - The following shows an example of this new keyword.

Oracle® Rdb for OpenVMS

```
$      define RDMS$SET_FLAGS "item_list,noprefix,strategy,detail(2)"
$      define RDMS$DEBUG_FLAGS_OUTPUT flags.log
$
$      RMU/UNLOAD-
          /OPTIMIZE=BITMAPPED_SCAN-
          RMU_UNLOAD_BITMAPPED_SCAN_4_DB-
          CURRENT_SALARY-
          CURRENT_SALARY
%RMU-I-DATRECUNL, 100 data records unloaded 5-AUG-2013 12:32:11.11.
$      SEARCH/REMAINING_FLAGS.LOG "~H Request"
~H Request Information Item List: (len=11)
RDB$K_SET_REQ_OPT_PREF "0"
RDB$K_SET_REQ_OPT_BITMAPPED "1"
RDB$K_INFO_END
Tables:
  0 = SALARY_HISTORY
  1 = EMPLOYEES
Cross block of 2 entries Q2
Cross block entry 1
  Leaf#01 BgrOnly 0:SALARY_HISTORY Card=729          Bitmapped scan
  Bool: MISSING (0.SALARY_END)
  BgrNdx1 SH_EMPLOYEE_SS [1:1] Fan=75
  Keys: MISSING (0.SALARY_END)
Cross block entry 2
  Get      Retrieval by index of relation 1:EMPLOYEES
  Index name EMP_EMPLOYEE_ID [1:1]      Direct lookup
  Keys: 1.EMPLOYEE_ID = 0.EMPLOYEE_ID
$
$      deassign RDMS$DEBUG_FLAGS_OUTPUT
$      deassign RDMS$SET_FLAGS
```

13.1.48 New EDIT STRING Clause for CREATE FUNCTION and CREATE MODULE Functions

This release of Oracle Rdb adds support for the EDIT STRING clause on a function definition. It allows the value returned from the function invocation to be implicitly formatted using the EDIT STRING associated with the function. This is similar to defining an EDIT STRING on a column or domain.

The EDIT STRING clause is only used by queries in Interactive SQL.

The following example shows a simple SQL function that returns the EMPLOYEE_ID but uses the EDIT STRING for Interactive SQL to add formatting.

```
SQL> create module SAMPLE
cont>     function SHOW_EMP_ID
cont>         (in :last_name varchar(30)
cont>           ,in :birthday date)
cont>     returns integer
cont>     edit string '9999'-'9999'-'99?'VOID"
cont>     ;
cont>     return
cont>         (select cast(employee_id as integer) * 100
cont>           from EMPLOYEES
cont>           where birthday = :birthday
cont>             and last_name = :last_name);
cont> end module;
SQL>
```

Oracle® Rdb for OpenVMS

```
SQL> select SHOW_EMP_ID (last_name, birthday), last_name, first_name
cont> from EMPLOYEES
cont> where employee_id < '00170';

          LAST_NAME      FIRST_NAME
0000-0164-00 Toliver      Alvin
0000-0165-00 Smith         Terry
0000-0166-00 Dietrich      Rick
0000-0167-00 Kilpatrick   Janet
0000-0168-00 Nash         Norman
0000-0169-00 Gray         Susan
6 rows selected
SQL>
SQL> -- demonstrate the output when a NULL result is returned
SQL> select SHOW_EMP_ID ('Unknown', current_date)
cont> from EMPLOYEES
cont> fetch first row only;

VOID
1 row selected
SQL>
```

The ALTER FUNCTION statement can be used to remove the edit string from a function (DROP EDIT STRING clause), or add/replace an edit string (EDIT STRING clause).

The "DROP EDIT STRING" clause removes any EDIT STRING that was previously defined for the function. No error is reported if there is no current edit string.

```
SQL> create function lib$lp_lines () returns integer;
cont> external language general
cont> general parameter style
cont> edit string 'S9(9)';
SQL> show function lib$lp_lines
Information for function LIB$LP_LINES

Function ID is: 4
Edit String:   S9(9)
Language is: GENERAL
GENERAL parameter passing style used
Number of parameters is: 0

Parameter Name          Data Type          Domain or Type
-----
                           INTEGER
Function result datatype
Return value is passed by value

SQL> alter function lib$lp_lines drop edit string;
```

13.1.49 Changes to RMU/VERIFY/CONSTRAINTS and ALTER TABLE Statement

With this release of Oracle Rdb, the RMU/VERIFY/CONSTRAINTS processing and the action of ALTER TABLE ... ENABLE ALL CONSTRAINTS has changed.

These changes include:

- ◆ PRIMARY KEY and UNIQUE constraints are now verified using a rewritten query that performs a single table scan. In the absence of a suitable index, the I/O required should be considerably reduced.
- ◆ NOT NULL constraints and the not null restriction for PRIMARY KEY constraints are validated using a single table scan for all such constraints on a table. This should reduce the table sequential scans in the absence of a suitable index for each constraint. Note that the side effect is that only the first failing constraint name is reported.
- ◆ When multiple tables are verified, the constraints are now ordered by table name. The goal is to make use of any buffered table rows for subsequent constraint queries. In prior versions, the constraints were verified in (approximately) definition order which might result in other tables being read and buffered data not being available.
- ◆ During RMU/VERIFY/CONSTRAINTS or during ALTER TABLE ... ENABLE ALL CONSTRAINTS, more detailed information on the verify can be seen by defining the ITEM_LIST flag. This can be done using either the SET FLAGS statement in SQL or defining the logical name RDMS\$SET_FLAGS.
In the case of a failing NOT NULL constraint, the DBKEY of the failing row is reported. The failure status (VMS condition code) is also reported along with the associated message text.
- ◆ The algorithms used by prior releases of Oracle Rdb remain available to RMU/VERIFY/CONSTRAINTS using the new FALLBACK keyword of the /CONSTRAINTS qualifier.

```

$ DEFINE/USER RDMS$SET_FLAGS ITEM_LIST
$ RMU/VERIFY/CONSTRAINTS=FALLBACK PERSONNEL
~H Extension (VERIFY CONSTRAINTS) Item List: (len=0)
~H: ...verify constraint "COLLEGE_CODE_REQUIRED"
~H: ...verify constraint "DEPT_CODE_REQUIRED"
~H: ...verify constraint "EMPLOYEE_ID_REQUIRED"
~H: ...verify constraint "JH_EMP_ID_EXISTS"
~H: ...verify constraint "JOB_CODE_REQUIRED"
~H: ...verify constraint "SH_EMP_ID_EXISTS"
~H: 6 tables processed.
$

```

13.1.50 New SQRT Numeric Function

The function SQRT returns the square-root of the passed value expression. If the expression is NULL then the result will be NULL. Only positive values can produce a square-root. Input values are converted to DOUBLE PRECISION, if necessary. The result of the function is a DOUBLE PRECISION value.

Note

Applications which call a user defined function with the same name will continue to do so if the name is delimited (for example "SQRT") or is part of an SQL Precompiler or SQL Module Language application compiled by a prior Rdb version. In other cases, Interactive and Dynamic SQL and applications compiled using Oracle Rdb Release 7.3.1 or later will use the new built-in function.

Syntax

```
--> SQRT ( --> value_expr --> ) ---->
```

Examples

The following examples show the result of using SQRT.

Example 13–3 Example 1: Invalid request for square root of a negative value

```
SQL> select SQRT (min (salary_amount) - max (salary_amount))
cont> from salary_history
cont> where employee_id < '00170'
cont> group by employee_id;
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-SYSTEM-F-FLTINV, floating invalid operation, PC=FFFFFFFF820E9362, PS=0000000B
SQL>
```

Example 13–4 Example 2: Correct query showing square root results

```
SQL> select SQRT (max (salary_amount) - min (salary_amount))
cont> from salary_history
cont> where employee_id < '00170'
cont> group by employee_id;

1.594396437527380E+002
6.772739475278819E+001
5.752390807307862E+001
5.009990019950140E+001
1.925486951396971E+002
9.897979591815695E+001
6 rows selected
SQL>
```

13.1.51 New MOD Numeric Function

The MOD function returns the remainder of the first value expression divided by the second value expression.

If either value expression is NULL, then the result will be NULL. The result of the function is either a DOUBLE PRECISION or BIGINT value. For ANSI and ISO SQL Dialects, the result type of the function is derived from the source argument types. Any floating point argument (REAL, DOUBLE PRECISION or FLOAT) will be reflected as a DOUBLE PRECISION result. Otherwise, a BIGINT result will be returned.

If the dialect is ORACLE LEVEL1, ORACLE LEVEL2, or ORACLE LEVEL3, then Oracle semantics allow MOD to return the value of the first argument if the second evaluates to zero. Otherwise, ANSI and ISO SQL Standard behavior results in a divide by zero exception being raised.

Note

Applications which call a user defined function with the same name will continue to do so if the name is delimited (for example "MOD") or is part of an SQL Precompiler or SQL Module Language application compiled by a prior Rdb version. In other cases, Interactive and Dynamic SQL and applications compiled

using Oracle Rdb Release 7.3.1 or later will use the new built-in function.

Syntax

```
--> MOD ( value_expr , value_expr ) -->
```

Examples

The following examples show the result of using MOD. This function uses MOD in the calculation of the days in a month.

Example 13–5 Example 1: Using the MOD function

```
SQL> drop module MOD_SAMPLE if exists;
SQL> create module MOD_SAMPLE
cont>
cont>     function DAYS_IN_MONTH (in :dt date)
cont>     returns integer
cont>     comment 'Compute days in the month of the given date'
cont>     ;
cont>     begin
cont>     declare :yr constant integer = extract (year from :dt);
cont>     declare :mo constant integer = extract (month from :dt);
cont>     return case :mo
cont>         -- 30 days has September, April, June, and November
cont>         when in (4,6,9,11) then 30
cont>         when 2 then
cont>             -- February has 28 unless it is a leap year
cont>             case MOD(:yr, 400)
cont>                 -- leap year if divisible by 400
cont>                 when 0 then 29
cont>                 else
cont>                     case MOD(:yr, 100)
cont>                         -- not a leap year if divisible by 100
cont>                         when 0 then 28
cont>                         else
cont>                             case MOD(:yr, 4)
cont>                                 -- leap year if divisible by 4
cont>                                 when 0 then 29
cont>                                 else 28
cont>                             end
cont>                         end
cont>                     end
cont>                 end
cont>             -- all the rest of 31 days
cont>             else 31
cont>         end;
cont>     end;
cont> end module;
```

13.1.52 New Data Types BINARY and BINARY VARYING

This release of Oracle Rdb adds two new data types BINARY and BINARY VARYING that allow definition of binary strings. These data types are specified by the ANSI and ISO SQL Language standard. These types would be suitable for storing small images, encrypted passwords, and so on.

Binary strings have the following characteristics:

- ◆ The SPACE octet for binary strings is X'00' (the zero valued octet). Therefore, when copying a BINARY string to a longer string it will be filled with X'00' and when comparing binary strings, the shorter string will be zero filled.
- ◆ The name VARBINARY is a synonym for BINARY VARYING.
- ◆ CONCAT (||), LIKE, MATCHING, OVERLAY, SUBSTRING, and TRIM operate on these types. The result data type of these operations will be a BINARY VARYING string. The clause USING { CHARACTERS | OCTETS } available for SUBSTRING and OVERLAY functions may not be used with BINARY or BINARY VARYING strings.
- ◆ POSITION, CHAR_LENGTH, and OCTET_LENGTH operate on binary string types. The CHAR_LENGTH function is equivalent to OCTET_LENGTH for these data types.
- ◆ CONTAINING, LIKE, MATCHING and STARTING WITH operate on binary string types but all input strings must be binary strings.
- ◆ Binary string literals can be specified using the X'hex-string' literal notation.

Note

In prior releases of Oracle Rdb, such literals inherited the LITERAL CHARACTER SET but this has changed to allow binary string assignment to UNSPECIFIED.

- ◆ When declaring host language variables in C or C++, the predefined \$SQL_VARBINARY should be used. This pseudo type creates a typedef in C that allows the length of the string to be passed to SQL, as frequently C zero terminated strings are inadequate to describe binary data that may need to embed X'00' values. The declared variable can reference the length (.len) and data (.data) as shown in the code sample below.

```
$SQL_VARBINARY(65000) sql_mem;
.
.
.
sql_mem.len = MAX_STRING;
memcpy(sql_mem.data,src_mem,sql_mem.len);
```

The resulting host variable will be type compatible with BINARY and BINARY VARYING columns and variables.

- ◆ Programmers can also use the CHARACTER SET BINARY clause to provide compatible host variables.

```
$SQL_VARCHAR(65000) CHARACTER SET BINARY sql_mem;
.
.
.
sql_mem.len = MAX_STRING;
memcpy(sql_mem.data,src_mem,sql_mem.len);
```

- ◆ Dynamic SQL applications will see the SQLTYPE field have the type SQLDA_BINARY (913) for BINARY expressions or SQLDA_VARBINARY (909) for BINARY VARYING expressions. The symbolic names are defined in SYSSHARE:SQL_LITERALS.H.

```
#define SQLDA_VARBINARY 909
```

```
#define SQLDA_BINARY 913
```

13.1.53 PERSONA SUPPORT is Enabled For All New Databases

In prior releases of Oracle Rdb, the CREATE DATABASE statement would not enable PERSONA SUPPORT by default. This meant that impersonation was done only using the OpenVMS UIC for the user. On the other hand, PERSONA SUPPORT uses the OpenVMS impersonation system services to impersonate the user including granted rights identifiers.

This release of Oracle Rdb changes the CREATE DATABASE statement to enable the PERSONA SUPPORT by default. This is shown below in this simple example.

```
SQL> create database
cont>     filename TESTING_DB;
SQL>
SQL> show database rdb$dbhandle;
Default alias:
    Oracle Rdb database in file TESTING_DB
.
.
.
    Shared Memory:          Process
    Large Memory:          Disabled
Security Checking is External (Persona support Enabled)
    System Index Compression is ENABLED
    System Index:
        Type is sorted ranked
        Prefix cardinality collection is enabled
    Logminer support is disabled
    Galaxy support is disabled
    Prestarted transactions are enabled
    Dictionary Not Required
    ACL based protections
Storage Areas in database with filename TESTING_DB
    RDB$SYSTEM              Default and list storage area
Journals in database with filename TESTING_DB
    No Journals found
Cache Objects in database with filename TESTING_DB
    No Caches found
SQL>
```

Generally when PERSONA SUPPORT is enabled, Rdb provides much better impersonation semantics for remote database access and for services such as SQL/Services and OCI Services for Rdb. However, this new default can be disabled using the PERSONA SUPPORT IS DISABLED clause for the ALTER DATABASE or CREATE DATABASE statement.

```
SQL> alter database filename TESTING_DB
cont> security checking is external (persona support is disabled);
SQL> attach 'filename TESTING_DB';
SQL> show database rdb$dbhandle
Default alias:
    Oracle Rdb database in file TESTING_DB
    Multischema mode is disabled
.
.
.
```

Oracle® Rdb for OpenVMS

```
Shared Memory:          Process
Large Memory:          Disabled
Security Checking is External
System Index Compression is ENABLED
System Index:
    Type is sorted ranked
    Prefix cardinality collection is enabled
Logminer support is disabled
Galaxy support is disabled
Prestarted transactions are enabled
Dictionary Not Required
ACL based protections
Storage Areas in database with filename TESTING_DB
    RDB$SYSTEM                Default and list storage area
Journals in database with filename TESTING_DB
    No Journals found
Cache Objects in database with filename TESTING_DB
    No Caches found
SQL>
```

13.1.54 New Dialects Support in SQL

This release of Oracle Rdb supports the following new dialects in SQL.

- ◆ ORACLE LEVEL3

This includes all the behavior described for ORACLE LEVEL2 plus the following changes:

- ◇ The same dialect rules as SQL2011 are in effect
- ◇ The DATE data type is assumed to mean ANSI style DATE which does not include time fields
- ◇ The CURRENT_TIMESTAMP builtin function returns a TIMESTAMP(2) type

- ◆ SQL2011

This includes all the behavior described for SQL99 plus the following changes:

- ◇ PRIMARY KEY or UNIQUE constraints must be evaluated at the same time or sooner than the referencing FOREIGN KEY constraints. That is, a FOREIGN KEY constraint defined as NOT DEFERRABLE may not reference a PRIMARY KEY or UNIQUE constraint defined as DEFERRABLE.

13.1.55 New WITH Clause Provides Subquery Factoring

This release of Oracle Rdb introduces the WITH clause as part of the SELECT expression. This feature, known as subquery factoring, allows the SQL programmer to simplify complex queries by creating named subqueries that can be used (possibly multiple times) in the associated SELECT expression.

The following example shows the declaration of two subquery factors, EMP and DPT, that are used in the select expression.

```
SQL> with emp as (select *
cont>                from employees inner join
cont>                job_history using (employee_id)
cont>                where job_end is null),
cont>    dpt as (select * from departments)
cont> select e.last_name, d.department_name, m.last_name as MANAGER
```

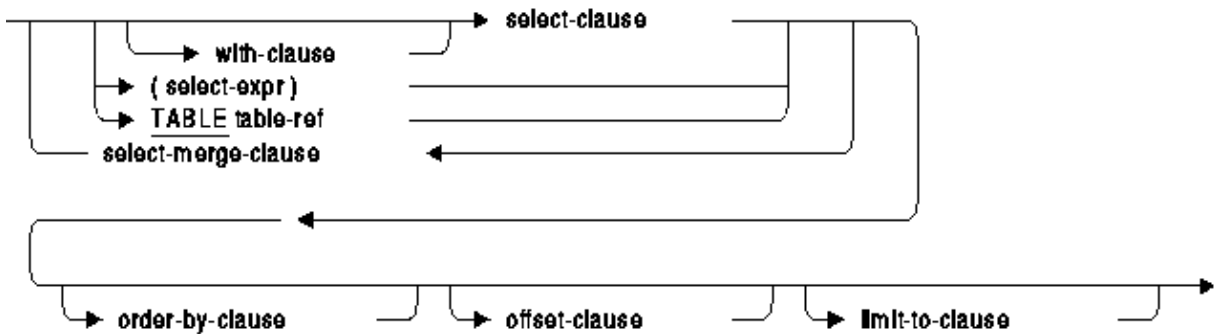
```

cont> from emp e
cont>         left outer join dpt d using (department_code)
cont>         inner join emp m on (d.manager_id = m.employee_id)
cont> order by d.manager_id
cont> fetch first row only
cont> ;
  E.LAST_NAME      D.DEPARTMENT_NAME      MANAGER
  Siciliano        Board Manufacturing North  Toliver
1 row selected
SQL>

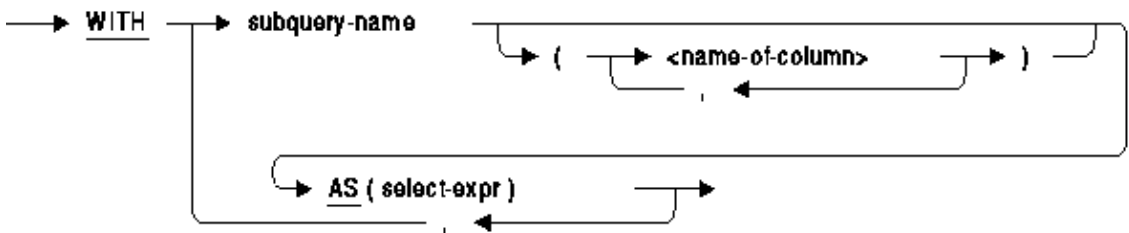
```

Syntax

select-expr =



with-clause =



Usage Notes

- ◆ The WITH clause can appear in any place that accepts a SELECT expression. For instance, when declaring a cursor, in an INSERT ... SELECT Statement, as part of Single SELECT Statement, as part of a nested subquery, a compound statement FOR loop, and so on.
- ◆ You may reuse the subquery name in separate queries, or in more deeply nested subqueries. You may not repeat the name in the same WITH clause. See the following example.

```

SQL> with
cont>     dept as (select * from departments where department_code <> 'PRES'),
cont>     dept as (select * from jobs)
cont> select count(*), (select department_name

```

Oracle® Rdb for OpenVMS

```
cont>          from dept
cont>          where jh.department_code = department_code)
cont> from job_history jh, dept d
cont> where jh.department_code = d.department_code
cont> group by jh.department_code
cont> ;
%SQL-F-DUPVAR, Variable DEPT is already defined
SQL>
```

- ◆ If you declare a subquery factor name but do not use it, an informational message will be issued by SQL. However, the query will still be executed.

```
SQL> with
cont>     dept as (select * from departments)
cont> select count(*), (select department_name
cont>                    from departments
cont>                    where jh.department_code = department_code)
cont> from job_history jh, departments d
cont> where jh.department_code = d.department_code
cont> group by jh.department_code
cont> ;
%SQL-I-VARNOTUSED, Variable "DEPT" was declared but never used
```

```
          15  Corporate Administration
          .
          .
          .
          12  Western U.S. Sales
26 rows selected
SQL>
```

- ◆ In prior versions of Oracle Rdb, it was permitted to follow the BEGIN keyword in a top level compound statement or stored routine with a WITH HOLD clause to specify that the procedure treated all FOR loops as HOLD cursors. Unfortunately this syntax conflicts with the WITH clause specified by the ANSI and ISO SQL Database Language Standard. Therefore, to accommodate this change, Oracle Rdb has removed the WITH HOLD syntax as a standalone clause after the BEGIN keyword. The alternate syntax, available since Oracle Rdb V7.1, is to use the PRAGMA clause which allows the WITH HOLD clause to be specified.

The following example shows the old syntax which now produces a syntax error message.

```
SQL>
SQL> begin
cont> with hold preserve none
      with hold preserve none
           ^
%SQL-W-LOOK_FOR_STT, Syntax error, looking for:
%SQL-W-LOOK_FOR_CON,          (, AS,
%SQL-F-LOOK_FOR_FIN,         found PRESERVE instead
```

It should be replaced with the following syntax which provides the same behavior.

```
SQL> begin
cont> pragma (with hold preserve none)
cont> trace 'a';
cont> end;
```

Examples

The following example shows the old syntax and the new syntax for the WITH clause. The old syntax causes a syntax error now.

Example 13–6 Example 1: Using the old syntax vs the new syntax for the WITH clause

```
SQL>
SQL> begin
cont> with hold preserve none
  with hold preserve none
      ^
%SQL-W-LOOK_FOR_STT, Syntax error, looking for:
%SQL-W-LOOK_FOR_CON,          (, AS,
%SQL-F-LOOK_FOR_FIN,    found PRESERVE instead

SQL> begin
cont> pragma (with hold preserve none)
cont> trace 'a';
cont> end;
```

This example shows the use of nested subquery factoring. The nested subqueries can in turn be factored.

Example 13–7 Example 2: Using Complex Query with INSERT ... SELECT Statement

```
SQL> declare local temporary table module.EMPS
cont>   like EMPLOYEES (job_count int, sal_count int);
SQL>
SQL> insert into module.EMPS
cont>   with emp_info as
cont>     (select e.*,
cont>         (with job_dept as
cont>           (select jh.department_code, jh.employee_id
cont>             from job_history jh
cont>             where jh.employee_id = e.employee_id)
cont>           select count(department_code) from job_dept),
cont>         (with sal_amt as
cont>           (select sh.salary_amount, sh.employee_id
cont>             from salary_history sh
cont>             where sh.employee_id = e.employee_id)
cont>           select count(salary_amount) from sal_amt)
cont>         from employees e)
cont>   select * from emp_info
cont> ;
100 rows inserted
SQL>
```

This query finds any other employee who started a new job on a significant date for other employees.

Example 13–8 Example 3: Using subquery factoring within a UNION operator

```
SQL> select e.last_name, jh.job_start
cont> from employees e, job_history jh
cont> where e.employee_id = jh.employee_id
cont> and jh.job_start in
cont>   (with
```

```

cont>          actual_jobs as
cont>          (select *
cont>             from job_history j
cont>             where j.job_end is null)
cont> select job_start from actual_jobs
cont>   where employee_id <> jh.employee_id
cont> union all
cont> with
cont>          actual_salary as
cont>          (select *
cont>             from salary_history s
cont>             where s.salary_end is null)
cont> select salary_start from actual_salary
cont>   where employee_id <> jh.employee_id)
cont> ;
E.LAST_NAME      JH.JOB_START
Kilpatrick      16-Aug-1980
Nash            17-Nov-1980
Danzig          2-Feb-1982
Gehr            9-Sep-1981
Clinton         28-May-1980
Siciliano       9-Sep-1981
Villari         16-Apr-1981
Jackson         3-Jan-1983
Gramby          28-May-1980
Flynn           2-Feb-1982
Flynn           1-Feb-1981
Keisling        3-Jan-1983
Klein           28-Dec-1980
Silver          7-Aug-1982
Belliveau       16-Apr-1981
Crain           28-Dec-1980
MacDonald       17-Nov-1980
17 rows selected
SQL>

```

13.1.56 DECLARE LOCAL TEMPORARY VIEW Statement

The DECLARE LOCAL TEMPORARY VIEW statement explicitly declares a local temporary view.

The metadata for a declared local temporary view is not stored in the database and cannot be shared by other modules.

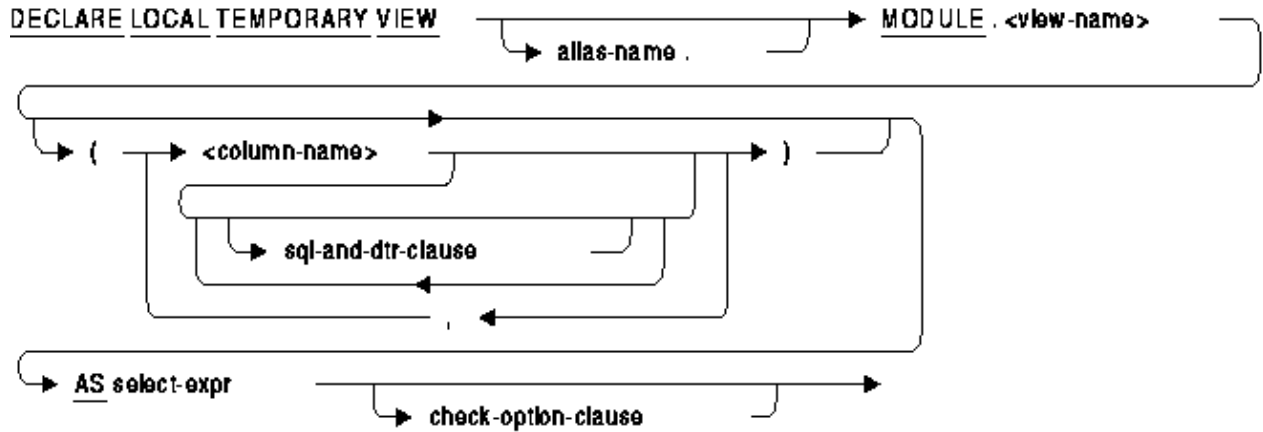
This statement allows an application to define view definitions that are temporary and do not require CREATE privilege on the database.

Environment

You can use the DECLARE LOCAL TEMPORARY VIEW statement:

- ◆ In interactive SQL
- ◆ In dynamic SQL as a statement to be dynamically executed
- ◆ In a stored module as part of the module header

Format



Usage Notes

- ◆ By using a declared view, queries using those views can be simplified.
- ◆ The view definition can specify QUERY HEADER and EDIT STRING, which are only used by Interactive SQL. If the temporary view is declared in a view, then these attributes of the column are ignored.
- ◆ The view definition can specify QUERY NAME and DEFAULT VALUE FOR DTR but these attributes of the column are ignored.
- ◆ A declared local temporary view acts like a created view. Refer to the CREATE VIEW Statement for further details.

Examples

The following example declares a view which is subsequently used in a SELECT statement. The QUERY HEADER and EDIT STRING are applied by the SELECT statement.

Example 13–9 Example 1: Simplifying a query using a declared local view

```
SQL> declare local temporary view module.employee_summary
cont> (eid
cont>     edit string 'XXBXXX'
cont>     comment is 'Employee id'
cont>     ,num_jobs
cont>     query name 'NUMBER_JOBS'
cont>     ,started
cont>     query header 'When'/'Started'
cont>     ,current_start
cont>     default value for dtr '1-Jan-1900 00:00:00.00')
cont> as select employee_id, count(*),
cont>             min (job_start), max (job_start)
cont>             from job_history
cont>             group by employee_id;
SQL>
SQL> select * from module.employee_summary where eid <= '00164';
      EID          NUM_JOBS   Started          CURRENT_START
```

```

00 164          2    5-JUL-1980    21-SEP-1981
1 row selected
SQL>

```

This example shows various operations on a local temporary view, including the definition of a CHECK OPTION constraint that prevents rows being inserted into the view that cannot also be retrieved by that view.

Example 13–10 Example 2: Operations on an updatable local view

```

SQL> declare local temporary view module.emp_name
cont>     (employee_id, last_name, first_name, middle_initial)
cont>     as select employee_id, last_name, first_name, middle_initial
cont>     from employees
cont>     where middle_initial is not null
cont>     with check option constraint OUT_OF_RANGE
cont> ;
SQL>
SQL> select * from module.emp_name;
EMPLOYEE_ID  LAST_NAME      FIRST_NAME     MIDDLE_INITIAL
00164        Toliver        Alvin          A
00165        Smith         Terry          D
.
.
.
00435        MacDonald     Johanna        P
00471        Herbener      James          Q
64 rows selected
SQL>
SQL> insert into module.emp_name values ('00001', 'Grey', 'Zane', NULL);
%RDB-E-INTEG_FAIL, violation of constraint OUT_OF_RANGE caused operation to fail
-RDB-F-ON_DB, on database RDB$DEFAULT_CONNECTION
SQL>
SQL> insert into module.emp_name values ('00001', 'Grey', 'Zane', 'A');
1 row inserted
SQL>
SQL> update module.emp_name
cont> set middle_initial = 'a'
cont> where middle_initial = 'A';
5 rows updated
SQL>
SQL> select * from module.emp_name where middle_initial = 'a';
EMPLOYEE_ID  LAST_NAME      FIRST_NAME     MIDDLE_INITIAL
00001        Grey           Zane           a
00164        Toliver        Alvin          a
00189        Lengyel       Peter          a
00229        Robinson      Tom            a
00416        Ames          Louie          a
5 rows selected
SQL>
SQL> rollback;

```

13.1.57 Enhancements for Buffered Read Support in SQL EXPORT DATABASE Command

This release of Oracle Rdb includes a new ROW COUNT clause as part of the EXPORT DATABASE Statement. EXPORT DATABASE now uses the buffered interface to reduce client/server exchanges while reading data rows from the source tables. In prior versions, each row was read one at a time. The default for ROW COUNT is 500 rows.

The database administrator can tune this value using the ROW COUNT clause demonstrated in the following example.

```
SQL> export database
cont>     filename MF_PERSONNEL
cont>     into SAVED_MFP
cont>     row count 1000
cont> ;
SQL>
```

13.1.58 New BITMAPPED SCAN Clauses Added to OPTIMIZE Clause

This release of Oracle Rdb allows the programmer to specify the clause OPTIMIZE FOR BITMAPPED SCAN as part of a query. This clause requests that the query optimizer attempt to use BITMAPPED SCAN if there exists multiple supporting indices in the query. The Rdb query optimizer may ignore this request if only one index is used or if no SORTED RANKED indices would be used to solve the query.

The following example shows the effect of using this new clause.

```
SQL> set flags 'strategy,detail(2)';
SQL>
SQL> select count(*)
cont>   from car
cont>   where make = 'holden'
cont>     and cyear = 1979
cont>     and colour = 'blue'
cont>     and (ctype = 'sedan' or ctype = 'wagon')
cont> optimize for bitmapped scan
cont> ;
Tables:
  0 = CAR
Aggregate: 0:COUNT (*) Q2
Leaf#01 BgrOnly 0:CAR Card=6047 Bitmapped scan
  Bool: (0.MAKE = 'holden') AND (0.CYEAR = 1979)
        AND (0.COLOUR = 'blue')
        AND ((0.CTYPE = 'sedan') OR (0.CTYPE = 'wagon'))
  BgrNdx1 IYEAR [1:1] Fan=97
    Keys: 0.CYEAR = 1979
  BgrNdx2 ICOLOUR [1:1] Fan=79
    Keys: 0.COLOUR = 'blue'
  BgrNdx3 IMAKE [1:1] Fan=79
    Keys: 0.MAKE = 'holden'
  BgrNdx4 ITYPE [(1:1)2] Fan=79
    Keys: r0: 0.CTYPE = 'wagon'
          r1: 0.CTYPE = 'sedan'

1
1 row selected
```

SQL>

In previous releases, the programmer would need to define the logical name `RDMS$ENABLE_BITMAPPED_SCAN` as 1, `RDMS$SET_FLAGS` as "BITMAPPED_SCAN", or use the `SET FLAGS 'BITMAPPED_SCAN'` statement in the application.

13.1.59 New Support for Allocations Specified Using Quantified Numeric Literal

This release of Oracle Rdb allows the database administrator to use allocation sizes using a quantified numeric literal. This shorthand notation allows the programmer to use numeric values that end in a multiplier represented by one of the following letters.

- ◆ K, meaning kilobytes
- ◆ M, meaning megabytes
- ◆ G, meaning gigabytes
- ◆ T, meaning terabytes
- ◆ P, meaning petabytes

The numeric value will be scaled according to the multiplier.

- ◆ If multiplier is K, then 1,024.
- ◆ If the multiplier is M, then 1,048,576.
- ◆ If the multiplier is G, then 1,073,741,824.
- ◆ If the multiplier is T, then 1,099,511,627,776.
- ◆ If the multiplier is P, then 1,125,899,906,842,624.

Note

Not all values specified by this notation are supported by the current release of Oracle Rdb.

These quantified numeric literals can be used with the following clauses and statements:

- ◆ **ALLOCATION** and **SNAPSHOT ALLOCATION** clause
As part of the **CREATE DATABASE** Statement or **IMPORT DATAABSE** Statement. These clauses provide the allocation for the default storage area `RDB$SYSTEM`, as well as the default allocations if none are specified for specific storage areas.
- ◆ **ALLOCATION** and **SNAPSHOT ALLOCATION** clause
As part of the **CREATE STORAGE AREA** clause, **ADD STORAGE AREA** clause, or **ALTER STORAGE AREA** clause. These clauses specify explicit sizes for new or altered storage area files.
- ◆ **ALLOCATION** clause
As part of the **CREATE**, **ADD** or **ALTER JOURNAL** clause. These clauses specify explicit allocation of the new journal file.
- ◆ **ALLOCATION** clause
As part of the **CREATE**, **ADD** or **ALTER CACHE** clause. These clauses specify explicit allocation of the caching backing file.
- ◆ **MEMORY ALLOCATION** clause
As part of the **GLOBAL BUFFERS** clause. This value specifies the amount of virtual

memory to allocate for the global buffers.

13.1.60 New SQL Functions Added

This release of Oracle Rdb adds new functions to the SYS\$LIBRARY:SQL_FUNCTIONS73.SQL script. To replace the existing library of functions, first use SQL_FUNCTIONS_DROP73.SQL script and then reapply using SQL_FUNCTIONS73.SQL.

Description

- ◆ BITANDNOT (numeric-expression, numeric-expression)
This function is used to clear bits in the first expression that are set in the second expression. First a bitwise NOT (BITNOT) is performed on the second numeric value expression and then a bitwise AND (BITAND) is performed of the first numeric value expression with the result.
If either of the passed expressions results in NULL then the result of BITANDNOT will be NULL. Note that BITANDNOT is equivalent to BITAND (exp1, BITNOT (ex2)) but is more efficient.
- ◆ BITNOT (numeric-expression)
Returns the bitwise NOT of the passed numeric value expression. If the passed expression results in NULL, then the result of BITNOT will be NULL.
- ◆ BITOR (numeric-expression, numeric-expression)
Returns the bitwise OR of the passed numeric value expressions. If either of the passed expressions results in NULL, then the result of BITOR will be NULL.
- ◆ BITXOR (numeric-expression, numeric-expression)
Returns the bitwise XOR of the passed numeric value expressions. If either of the passed expressions results in NULL, then the result of BITXOR will be NULL.

13.1.61 Optimized NOT NULL Constraint Execution

This release of Oracle Rdb introduces a new mechanism to verify NOT NULL constraints which are executed immediately at statement end (that is NOT DEFERRABLE). This new mechanism is more efficient (uses less code and virtual memory) than mechanisms used in prior releases. The cost of the constraint check in these cases is a fixed cost with a very small incremental cost for each extra NOT NULL constraint. The NOT NULL requirement of PRIMARY KEY constraints are also checked in the same way.

In prior releases of Oracle Rdb, each NOT NULL constraint would require its own internal query and each would be evaluated serially against the row just inserted or updated.

The following example shows an INSERT into a simple table with STRATEGY flags enabled. As can be observed, the absence of the strategy display indicates that no optimized query was used to validate these constraints.

```
SQL> set flags 'strategy,detail(2),internal,request_name';
SQL>
SQL> insert into SAMPLE
cont>      default values;
%RDB-E-INTEG_FAIL, violation of constraint SAMPLE_PK caused operation to fail
-RDB-F-ON_DB, on database RDB$DEFAULT_CONNECTION
SQL>
```

```

SQL> insert into SAMPLE (iden)
cont>      values (0);
%RDB-E-INTEG_FAIL, violation of constraint SAMPLE_DAT_NOT_NULL caused operation
to fail
-RDB-F-ON_DB, on database RDB$DEFAULT_CONNECTION
SQL>
SQL> insert into SAMPLE
cont>      values (1, 'A');
~Sn: Constraint "SAMPLE_PK" evaluated (verb)
Tables:
  0 = SAMPLE
  1 = SAMPLE
Cross block of 2 entries  Q1
Cross block entry 1
  Conjunct: 0.DBKEY = <var0>
  Firstn: 1
  Get      Retrieval by DBK of relation 0:SAMPLE
Cross block entry 2
  Conjunct: <agg0> <> 1
  Aggregate-F2: 0:COUNT-SINGLE (<subselect>) Q2
  Index only retrieval of relation 1:SAMPLE
    Index name  SAMPLE_NDX [1:1]
    Keys: 0.IDEN = 1.IDEN
1 row inserted
SQL>

```

Note that any DEFERRABLE constraints will be executed as in prior versions.

13.1.62 New RMU/LOAD Option CHARACTER_ENCODING_XML

When using RMU/UNLOAD/RECORD_DEFINITION/FORMAT=XML, the XML header record will, by default, use the character encoding "ISO-8859-1", as seen in the following example.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

This encoding (ISO-8859-1) is Latin 1 and covers encoding of many European character sets. However, this encoding is not adequate if you use other character encoding for Asian languages or languages not covered by this ISO Standard.

This release of Oracle Rdb adds a new option, CHARACTER_ENCODING_XML, that allows the command procedure to specify an alternate character encoding. For example, if the data being unloaded is using the UTF8 character set, use this new option as shown in this example.

```

$ rmu/unload-
  /record=(nofile,format=xml,trim,character_encoding_xml="utf-8")-
  sql$database -
  employees -
  employees
%RMU-I-DATRECUNL,    100 data records unloaded  8-SEP-2013 22:21:49.54.
$

```


13.1.63 New MEMORY ALLOCATION Clause for the GLOBAL BUFFERS Definition

This release of Oracle Rdb allows the database administrator to define the size of the GLOBAL BUFFER pool using direct memory sizing as an alternate to specifying the NUMBER of pages.

◆ **MEMORY ALLOCATION IS <mem-octets>**

The value of mem-octets is an unsigned numeric literal or a quantified numeric literal. This clause is not compatible with the NUMBER IS clause. Use just one of the keywords to define the size of the global buffers.

The following example shows the use of MEMORY ALLOCATION when creating global buffers.

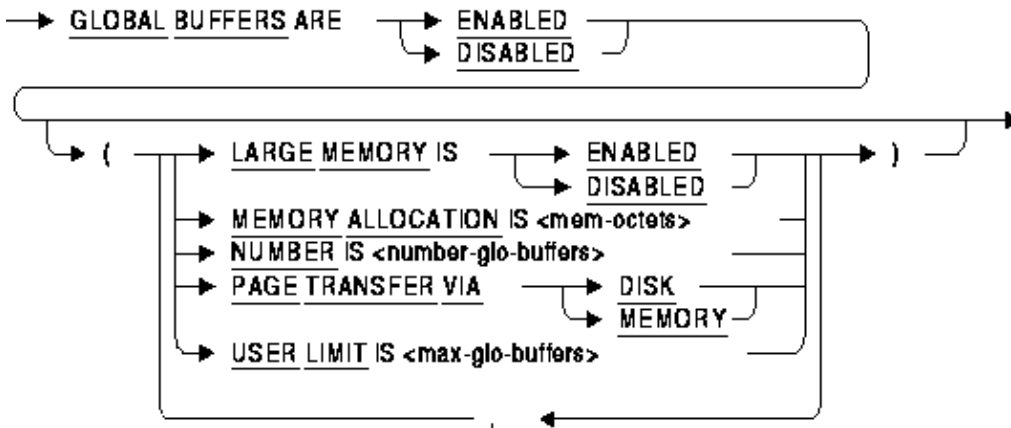
```
create database
  filename TEST_DB

  allocation 110k pages
  snapshot allocation 1k pages
  global buffers are disabled (memory allocation 1m)

create storage area AREAl
  allocation 100k pages
  snapshot allocation 2k pages
;
```

Syntax

global-buffer-params=



13.1.64 New REPLACE Statement

Bug 8929218

Oracle® Rdb for OpenVMS

This release of Oracle Rdb introduces a new REPLACE statement. When a table includes a PRIMARY KEY definition, the REPLACE statement uses the key information to remove the existing matching row prior to inserting the replacement data.

The following example shows an example of the REPLACE statement. Triggers are defined with only TRACE statements to show the order of execution during REPLACE.

```
SQL> set dialect 'sql2011';
SQL> set flags 'test_system';
SQL>
SQL> create table SAMPLE
cont>     (ident integer primary key
cont>     ,description char(40)
cont>     );
SQL>
SQL> create trigger AI_SAMPLE
cont>     after insert on SAMPLE
cont>     (trace 'after an insert')
cont>     for each row;
SQL>
SQL> create trigger BI_SAMPLE
cont>     before insert on SAMPLE
cont>     (trace 'before an insert')
cont>     for each row;
SQL>
SQL> create trigger AD_SAMPLE
cont>     after delete on SAMPLE
cont>     (trace 'after a delete')
cont>     for each row;
SQL>
SQL> create trigger BD_SAMPLE
cont>     before delete on SAMPLE
cont>     (trace 'before a delete')
cont>     for each row;
SQL>
SQL> set flags 'trace';
SQL>
SQL> -- first row
SQL> insert into SAMPLE
cont>     values (100, 'First description');
~Xt: before an insert
~Xt: after an insert
1 row inserted
SQL>
SQL> -- should fail (duplicate)
SQL> insert into SAMPLE
cont>     values (100, 'Second description');
~Xt: before an insert
~Xt: after an insert
%RDB-E-INTEG_FAIL, violation of constraint SAMPLE_PRIMARY_IDENT caused
operation to fail
-RDB-F-ON_DB, on database RDB$DEFAULT_CONNECTION
SQL>
SQL> replace into SAMPLE
cont>     values (100, 'Replace first description');
~Xt: before a delete
~Xt: after a delete
~Xt: before an insert
~Xt: after an insert
1 row replaced
SQL>
```

```
SQL> select * from SAMPLE order by ident;
      IDENT  DESCRIPTION
      ----  -
      100    Replace first description
1 row selected
SQL>
SQL> commit;
SQL>
```

Usage Notes

- ◆ If no PRIMARY KEY exists for the table, or it is disabled, the REPLACE statement acts exactly like an INSERT statement.
- ◆ REPLACE is a valid statement for a TRIGGER action.
- ◆ In addition to the BEFORE and AFTER INSERT triggers, REPLACE will cause BEFORE and AFTER DELETE triggers to execute.
- ◆ REPLACE is a valid compound-use statement and can be used in a stored procedure.
- ◆ It is possible that the implicit DELETE action taken by REPLACE will cause constraint execution. These constraints may prevent the DELETE actions (due to a table dependency) and therefore cause the REPLACE to fail.

13.1.65 Query Optimization Improvements for IN Clause

Bugs 12548885 and 14471918

The EXISTS and IN predicates can often be used interchangeably in queries to check for the existence of values in another result set. If possible, the EXISTS query should be the first preference because its structure allows for the best query optimization. However, the semantics of these predicates are not identical when NULL values are present in one or both tables, especially when used with the NOT operator. Care should be taken to ensure correct query behavior in such cases.

With this release of Oracle Rdb, the optimizer will attempt to transform the IN predicate to an EXISTS predicate when the source columns are known to be not nullable. Such a transformation will return the same results and additionally present a better query for optimization.

The following example shows that the strategy selected for NOT IN when the optimization is not (or cannot be) applied.

```
SQL> select s.badge_number
cont> from STAFF s
cont> where s.badge_number NOT IN (select kb.badge_number from KNOWN_BADGES kb)
cont> ;
Tables:
  0 = STAFF
  1 = KNOWN_BADGES
Cross block of 2 entries  Q1
Cross block entry 1
  Index only retrieval of relation 0:STAFF
  Index name  STAFF_I [0:0]
Cross block entry 2
  Conjunct: <agg0> = 0
  Aggregate-F1: 0:COUNT-ANY (<subselect>) Q2
  Conjunct: MISSING (0.BADGE_NUMBER) OR MISSING (1.BADGE_NUMBER) OR (
            0.BADGE_NUMBER = 1.BADGE_NUMBER)
  Index only retrieval of relation 1:KNOWN_BADGES
  Index name  KNOWN_BADGES_I [0:0]
```

```
BADGE_NUMBER
      4
1 row selected
SQL>
```

When the target columns (for example BADGE_NUMBER) in each table have a NOT DEFERRABLE constraint of the type PRIMARY KEY or NOT NULL, then the following strategy is used. The resulting strategy will likely result in faster query execution.

```
SQL> select s.badge_number
cont> from STAFF s
cont> where s.badge_number NOT IN (select kb.badge_number from KNOWN_BADGES kb)
cont> ;
Tables:
  0 = STAFF
  1 = KNOWN_BADGES
Conjunct: <agg0> = 0
Match    (Agg Outer Join) Q1
Outer loop
Match_Key:0.BADGE_NUMBER
  Index only retrieval of relation 0:STAFF
    Index name  STAFF_I [0:0]
Inner loop    (zig-zag)
Match_Key:1.BADGE_NUMBER
Index_Key:BADGE_NUMBER
  Aggregate-F1: 0:COUNT-ANY (<subselect>) Q2
    Index only retrieval of relation 1:KNOWN_BADGES
      Index name  KNOWN_BADGES_I [0:0]
BADGE_NUMBER
      4
1 row selected
SQL>
```

This transformation is enabled by default but can be disabled using SET FLAGS 'NOERWRITE(IN_CLAUSE)' and re-enabled using SET FLAGS 'REWRITE(IN_CLAUSE)'.

This new feature was introduced in Oracle Rdb Release 7.3.1.0.

Chapter 14

Documentation Corrections, Additions and Changes

This chapter provides corrections for documentation errors and omissions.

14.1 Documentation Corrections

14.1.1 Oracle Rdb Release 7.3.x.x New Features Document Added

A new document has been created which contains all of the New Features Chapters from all Rdb 7.3 Release Notes. This document will be included in saveset A of the Rdb kit. It is called RDB_NEWFEATURES_73xx 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 7.3 releases.

14.1.2 Undocumented /TRANSACTION=EXCLUSIVE Option for RMU Populate_Cache

The EXCLUSIVE option was not documented in prior releases of Oracle Rdb. This is a revised section for the RMU Populate_Cache Command and will appear in a future update to the Oracle RMU Reference Manual.

- ◆ Transaction_Type=option
Allows you to specify the transaction mode for the transactions used to perform the populate cache operation. Valid options are:
 - ◇ Automatic
 - ◇ Exclusive
 - ◇ Read_Only
 - ◇ Noread_Only

You must specify an option if you use this qualifier.

If you do not specify this qualifier, the default is Transaction_Type=Automatic. This qualifier specifies that Oracle RMU is to determine the transaction mode. For example, if READ ONLY transactions can not be performed, an equivalent READ WRITE transaction will be started.

The Transaction_Type=Read_Only qualifier specifies the transactions used to perform the analyze operation be set to read-only mode.

The Transaction_Type=Noread_Only qualifier specifies that the transactions used for the analyze operation be set to read/write mode. You might select this option if you want to avoid the growth of snapshot files that occurs during a long running read-only transaction and are willing to incur the cost of increased locking that occurs during a read/write transaction.

The Transaction_Type=Exclusive readies all accessed storage areas in EXCLUSIVE mode. Therefore, it avoids the growth of snapshot files (no updates will be made to any snapshot file) and will use limited locking.

Note

Using Transaction_Type=Exclusive might interfere with other actions occurring on the database; especially note that ONLINE backup and some actions of the Row Cache server are incompatible with this Transaction_Type.

14.1.3 Action of DECLARE TRANSACTION When SQL\$DATABASE is Defined

Bug 5705204

The following clarification is missing from the DECLARE TRANSACTION Statement documentation in the Oracle Rdb SQL Reference Manual.

- ◆ In Dynamic SQL and Interactive SQL, the DECLARE TRANSACTION statement is applied to all attached databases. When those databases are disconnected, the declared default transaction is discarded. If there are no attached databases, then the DECLARE TRANSACTION is used as a session wide default.

When the first executable statement in a Dynamic SQL or an Interactive SQL session is a DECLARE TRANSACTION statement and the logical name SQL\$DATABASE is defined, then the DECLARE statement will implicitly attach to that database. The declared default transaction is applied to that database attach. A subsequent DISCONNECT statement will disconnect from the database and discard the default transaction.

14.1.4 RDB\$USAGE Field Values

The following information is missing from the Rdb SQL Reference Manual.

The table Rdb\$INTERRELATIONS records much of the dependency information when one object references another in the database. Such information is used by DROP ... RESTRICT statements to prevent an object being deleted when it is required by some other object. For instance, a function may use one or more columns from a table in a query. That table and its columns will be recorded with a value in RDB\$USAGE of 'Storage Map'.

Many reported errors include text from the RDB\$USAGE field to explain the type of dependency preventing the DROP from succeeding. These text strings are described in [Table 14–1. Rdb\\$USAGE Field Values](#).

Table 14–1 Rdb\$USAGE Field Values

Field Value	Description
Computed Field	A Computed by or Automatic column references this table, view, column or function
Constraint	Constraint definition references table, view, column, sequence or function
Storage Map	Storage map references table and column
View	View definition requires table, view, column, sequence or function
View Field	View column requires table, view, column, sequence or function
Trigger	Trigger definition requires table, view, column, sequence or function
RelConstraint	A table (relation) constraint references a table
	A domain constraint (or VALID IF) references this routine or sequence

Domain Constraint (VALID IF)	
Requires	This table, temporary table (with module name), or index is used by a query outline
Procedure	Procedure definition requires table, view, column, sequence or function
Function	Function definition requires table, view, column, sequence or function
Default Txn Reserving	A stored module DECLARE TRANSACTION references a table or view in the RESERVING clause
Default Txn Evaluating	A stored module DECLARE TRANSACTION references a constraint in the EVALUATING clause
Lang Semantics	A stored function, procedure or trigger uses wildcard for column list. This includes SELECT *, or INSERT with an omitted column list
Cast As Domain	A CAST function referenced a domain name
Temp Table Using Domain	A DECLARE LOCAL TEMPORARY TABLE used a domain for a columns data type
Computed Column in Temp Table	A computed by or automatic column defined by a DECLARE LOCAL TEMPORARY TABLE or DECLARE LOCAL TEMPORARY VIEW references this object
Module Variable Default Value	A module global DECLARE statement used a DEFAULT clause. Table, view, function, domain and sequence dependencies are recorded
Referenced by Synonym	When a synonym is created, a dependency is stored
Default Value	A table column uses a DEFAULT clause. Table, view, function, domain and sequence dependencies are recorded
Constraint Index	When SET FLAGS 'AUTO_INDEX' is active, any constraint definition will define an index matching the columns of the constraint
Module Variable	This module variable uses this domain
Routine Parameter	Not currently used. Reserved for future use
Temp Table Reference	A DECLARE LOCAL TEMPORARY TABLE references this table in the LIKE clause
Storage Map Function	When CREATE STORAGE MAP is executed, a system routine is created to reflect the mapping. Those column dependencies are recorded

14.1.5 Updated Documentation for RDMS\$BIND_CODE_OPTIMIZATION Logical

The logical name RDMS\$BIND_CODE_OPTIMIZATION has different uses on Alpha and Integrity systems.

◆ Alpha Systems

In prior releases of Oracle Rdb, queries and procedures were rendered as small Alpha instruction sequences which were used at runtime to implement all or part of the query strategy. This meant that executable code and data could reside on any Alpha page (8192 byte) sections.

Recent performance testing shows that OpenVMS for Alpha running on an Alpha emulator

(such as Stromasys, Inc's CHARON-AXP) can greatly benefit from isolating the generated code instructions from updatable data areas. This change in memory management can be enabled by defining this logical to the value 3. Please note that such systems will typically require more virtual memory for the alternate memory management.

This logical name can be defined to the value (0) to disable the alternate memory management or defined as (3) to enable the alternate memory management. The default is 0 on Alpha systems and other values are undefined.

For more information on Stromasys, Inc, please check their web site at:

<https://www.stromasys.com/solutions/charon-axp/>

◆ Integrity systems

When Oracle Rdb first became available on the Integrity platform, it used a portable instruction interpreter. Now, by default, Oracle Rdb generates native IA64 instructions to implement part of the query and procedure execution code. This native code execution in general delivers improved performance but at the expense of more virtual memory usage. This logical name can be defined to the value (0) to disable the native instruction compiler and use the interpreted instructions, or defined as (2) to enable the native IA64 instruction generator. The default is 2 on Integrity systems and other values are undefined.

The RMU Show Logical_Names command can be used to display the latest documentation for this and other logical names used by Oracle Rdb.

```
$ rmu /show Logical_Names RDM$BIND_CODE_OPTIMIZATION/Description
  "RDM$BIND_CODE_OPTIMIZATION" = Undefined
...etc...
```

14.1.6 New RDM\$BIND_AIJ_BUFFER_POOL_COUNT Logical

The description of this new logical was inadvertently left out of the Rdb Release 7.3.1.3 Release Notes.

A new RDM\$BIND_AIJ_BUFFER_POOL_COUNT logical has been added for use by various components of Oracle Rdb. This logical name can be used to change the number of buffers in the internal buffer pool used by the RMU/UNLOAD AFTER_JOURNAL command to select and process journal records from database transactions. If the logical name is not defined, a default of 10 buffers will be used.

The RDM\$BIND_AIJ_BUFFER_POOL_COUNT logical allows a whole integer number of buffers between 4 and 16 to be specified for the RMU/UNLOAD AFTER_JOURNAL buffer pool. The current value of the RDM\$BIND_AIJ_BUFFER_POOL_COUNT logical can be displayed using the RMU/SHOW LOGICAL_NAMES command if it is defined as a system logical.

The following examples show that the RMU/SHOW LOGICAL_NAMES command can be used to show if RDM\$BIND_AIJ_BUFFER_POOL_COUNT is currently defined as a system logical. They also show that RDM\$BIND_AIJ_BUFFER_POOL_COUNT can be defined as a process logical, and that if a value is defined for RDM\$BIND_AIJ_BUFFER_POOL_COUNT that is not a whole integer between 4 and 16, an error will be returned by the RMU/UNLOAD AFTER_JOURNAL command and the RMU/UNLOAD AFTER_JOURNAL command will be aborted without processing any After Image Journal records.

Oracle® Rdb for OpenVMS

```
$ RMU/SHOW LOGICAL_NAMES RDM$BIND_AIJ_BUFFER_POOL_COUNT
$
$!
$ RMU/SHOW LOGICAL_NAMES/UNDEFINED RDM$BIND_AIJ_BUFFER_POOL_COUNT
  "RDM$BIND_AIJ_BUFFER_POOL_COUNT" = Undefined
$!
$ DEFINE/SYSTEM RDM$BIND_AIJ_BUFFER_POOL_COUNT 4
$ RMU/SHOW LOGICAL_NAMES RDM$BIND_AIJ_BUFFER_POOL_COUNT
  "RDM$BIND_AIJ_BUFFER_POOL_COUNT" = "4" (LNM$SYSTEM_TABLE)
$!
$ DEFINE/SYSTEM RDM$BIND_AIJ_BUFFER_POOL_COUNT 16
%DCL-I-SUPERSEDE, previous value of RDM$BIND_AIJ_BUFFER_POOL_COUNT has been
superseded
$ RMU/SHOW LOGICAL_NAMES RDM$BIND_AIJ_BUFFER_POOL_COUNT
  "RDM$BIND_AIJ_BUFFER_POOL_COUNT" = "16" (LNM$SYSTEM_TABLE)
$!
$ DEFINE RDM$BIND_AIJ_BUFFER_POOL_COUNT 3
$ RMU/UNLOAD/AFTER_JOURNAL/INCL=ACT=(NOCOMMIT)/STAT=3600/LOG -
/RESTORE_METADATA=LOGMINER_METADATA.TXT -
/TABLE=(NAME=TEST_TABLE,OUTPUT=RDB_LOGMINER_OUTPUT_FILE) -
TEST.AIJ
%RMU-I-LMMFRDCNT, Read 13542 objects from metadata file
"DEVICE:[DIRECTORY]LOGMINER_METADATA.TXT;1"
%RMU-I-UNLAIJFL, Unloading table TEST_TABLE to
DEVICE:[DIRECTORY]RDB_LOGMINER_OUTPUT_FILE.DAT;1
%RMU-F-BADBNDPRM, bad bind parameter RDM$BIND_AIJ_BUFFER_POOL_COUNT value "3"
-RMU-F-BADBOUNDS, value not in range 4 to 16
%RMU-F-FTL_RMU, Fatal error for RMU operation at 17-MAR-2019 09:58:08.42
-----
  ELAPSED:    0 00:00:00.00  CPU: 0:00:00.00  BUFIO: 5  DIRIO: 0  FAULTS: 46
Table "TEST_TABLE" : 0 records written (0 modify, 0 delete)
Total : 0 records written (0 modify, 0 delete)
$!
$ DEFINE RDM$BIND_AIJ_BUFFER_POOL_COUNT 17
%DCL-I-SUPERSEDE, previous value of RDM$BIND_AIJ_BUFFER_POOL_COUNT has been
superseded
$ RMU/UNLOAD/AFTER_JOURNAL/INCL=ACT=(NOCOMMIT)/STAT=3600/LOG -
/RESTORE_METADATA=LOGMINER_METADATA.TXT -
/TABLE=(NAME=TEST_TABLE,OUTPUT=RDB_LOGMINER_OUTPUT_FILE) -
TEST.AIJ
%RMU-I-LMMFRDCNT, Read 13542 objects from metadata file
"DEVICE:[DIRECTORY]LOGMINER_METADATA.TXT;1"
%RMU-I-UNLAIJFL, Unloading table TEST_TABLE to
DEVICE:[DIRECTORY]RDB_LOGMINER_OUTPUT_FILE.DAT;2
%RMU-F-BADBNDPRM, bad bind parameter RDM$BIND_AIJ_BUFFER_POOL_COUNT value "17"
-RMU-F-BADBOUNDS, value not in range 4 to 16
%RMU-F-FTL_RMU, Fatal error for RMU operation at 17-MAR-2019 09:59:39.02
-----
  ELAPSED:    0 00:00:00.00  CPU: 0:00:00.01  BUFIO: 5  DIRIO: 0  FAULTS: 46
Table "TEST_TABLE" : 0 records written (0 modify, 0 delete)
Total : 0 records written (0 modify, 0 delete)
$!
```

14.1.7 RDMSTT Image Optionally Installed

Bug 3981903

If you plan on using the cluster capability of RMU/SHOW STATISTICS, Oracle recommends that you install the RDMSTT73.EXE image on OpenVMS with the appropriate privileges.

The RMONSTART73.COM command procedure provided by the Oracle Rdb installation allows the system manager to optionally install the RDMSTT73.EXE image at monitor startup time. To take advantage of this, you will need to edit the SYS\$STARTUP:RMONSTART73.COM procedure and remove the comment characters from the following lines:

```
$ ! DEFN SYS$COMMON:[SYSEXE]RDMSTT73.EXE
$ !      REMOVEX
$ !      ADDX /OPEN/HEAD/PROT/PRIV=(CMKRNL,SYSPRV,SHARE)
```

Also, edit the command procedure SYS\$STARTUP:RMONSTOP73.COM and remove the comment characters from the following lines:

```
$ ! DEFN SYS$COMMON:[SYSEXE]RDMSTT73.EXE
$ !      REMOVEX
```

14.1.8 Clarification on the Affect of the RMU/OPEN/ACCESS=RESTRICTED Command

The command RMU/OPEN/ACCESS=RESTRICTED can be used to prevent non-privileged users attaching to the database while database maintenance is being performed.

This command sets a permanent attribute in the database root that must be cleared by a subsequent RMU/OPEN/ACCESS=UNRESTRICTED command. This is true even if the database is defined OPEN IS AUTOMATIC. An RMU/CLOSE might be required before the RMU/OPEN can be executed.

The following example shows the reported error when this attribute is active; NOPRIV with an explanation DBNOTOPEN.

```
SQL> att 'f personnel user 'dbuser2' using '*****' ;
%SQL-F-ERRATTDEC, Error attaching to database personnel
-RDB-E-NO_PRIV, privilege denied by database facility
-RDMS-F-DBNOTOPEN, database is not open for access
-RDB-F-ON_DB, on database _$1$DGA174:[TESTING]PERSONNEL.RDB;1
SQL>
```

The database administrator must be granted RMU\$OPEN on the database root file, as well as having DBADM on the database (this might be inherited from OpenVMS override privileges) to execute this command.

The RMU/DUMP/HEADER command will show this setting as seen in the following output:

```
$ rmu/dump/header=ROOT_RECORD PERSONNEL
*-----*
* Oracle Rdb V7.3-210                                22-NOV-2016 14:27:23.59
*
* Dump of Database header
*      Database: USER2:[TESTING]PERSONNEL.RDB;1
*
*-----*

Database Parameters:
      Root filename is "USER2:[TESTING]PERSONNEL.RDB;1"
```

Oracle Rdb specific root record

```
Dbkey for Oracle Rdb bootstrap page is 8:462:0
Access restricted to privileged users
Current metadata version is 26
Database has never been backed up
Database has never been incrementally restored
Database has never been fully restored
Database has never been verified
Database has never been altered
```

14.1.9 Some Optional System Tables Can Be Relocated Using a User Defined Storage Map

All system tables are mapped by default to the system storage area RDB\$SYSTEM. If the database is created with the DEFAULT STORAGE AREA clause, then some of these tables will automatically be created in the secondary system area. Additionally, there exists a set of optional system tables (which may not exist in all databases) which may be manually mapped to other storage areas.

To change the mapping for one (or more) of these system tables, you must follow these steps. See [Table 14–2](#) for the list of these optional system tables that allow mapping.

1. Attach to the database. If you are creating a storage map for RDB\$CATALOG_SCHEMA or RDB\$SYNONYMS then you must attach with the option MULTISchema IS OFF. You should not execute any queries on the database as this may cause the system table to be locked and prevent the CREATE and ALTER STORAGE MAP statements from completing.
2. Create a storage map for the optional system table. Note that only those listed here are able to be re-mapped and you must use the specified storage map names.

Table 14–2 Optional System Tables and Their Storage Map Names

Table Name	Storage Map Name	Associated Feature
RDB\$CATALOG_SCHEMA	RDB\$CATALOG_SCHEMA_MAP	Multischema databases
RDB\$CHANGES	RDB\$CHANGES_MAP	Replication Option for Rdb
RDB\$CHANGES_MAX_TSER	RDB\$CHANGES_MAX_TSER_MAP	Replication Option for Rdb
RDB\$SYNONYMS	RDB\$SYNONYMS_MAP	Multischema databases
RDB\$TRANSFERS	RDB\$TRANSFERS_MAP	Replication Option for Rdb
RDB\$TRANSFER_RELATIONS	RDB\$TRANSFER_RELATIONS_MAP	Replication Option for Rdb
RDB\$WORKLOAD	RDB\$WORKLOAD_MAP	Workload Collection was

3. The storage map must be a simple storage map which simply describes the current state for this table, namely the name of the storage area in which the table resides. See the following example.

```
SQL> create storage map RDB$CHANGES_MAP
cont>   for RDB$CHANGES
cont>   store in RDB$SYSTEM;
```

The following restrictions apply to the created storage map for these special system tables:

- ◇ The storage map may not change the defaulted compression attributes
 - ◇ The storage map may not specify the logical area thresholds
 - ◇ The storage map may not be placed via an index
 - ◇ The storage map may not vertically partition the table
 - ◇ The storage map may only contain one storage area
 - ◇ And it must be mapped to the default storage area (this may be RDB\$SYSTEM by default or the name of the user specified storage area using the DEFAULT STORAGE AREA clause during CREATE DATABASE)
4. Now that the storage map exists, you may use the ALTER STORAGE MAP statement to move the table to another area.

```
SQL> alter storage map RDB$CHANGES_MAP
cont>   store in RDB_CHANGES_AREA;
```

The following restrictions apply to the altered storage map for these special system tables:

- ◇ The storage map may not be placed via an index
 - ◇ The storage map may only contain one storage area
 - ◇ The storage map may not vertically partition the table
 - ◇ The ALTER STORAGE MAP operation may require exclusive access to the database as well as the table while the table data is relocated.
5. These storage map attributes for system tables are not currently exported by the SQL EXPORT DATABASE statement. Therefore, if you EXPORT and IMPORT your database, you will need to repeat these steps to re-map any of these system tables. It is expected that this restriction will be removed in a future version of Oracle Rdb.

14.1.10 Logical Name RDM\$BIND_HOT_NETWORK_OBJECT Was Not Documented

Bug 2534163

The following logical name was not previously documented for Oracle Rdb.

- RDM\$BIND_HOT_NETWORK_OBJECT

The name of the network object for the AIJSERVER process on the remote standby database. This

logical can be used with the following transports: DECnetIV, DECnetOSI, and HPE TCP/IP Services for OpenVMS.

Specify RDMAIJ<version-number> for Oracle Rdb databases (example: RDMAIJ73).

Please note that the RMU/SHOW LOGICAL/UNDEFINED/DESCRIPTION command will display the description for this logical name.

14.1.11 Recovering an Oracle Rdb Database After RMU/RESTORE/ONLY_ROOT

Bug 12595718

If it is necessary to recover a database following the RMU/RESTORE/ONLY_ROOT command, be sure that the transaction state of the database is correctly set in the database root by the RMU/RESTORE/ONLY_ROOT command. Otherwise transactions in journal files created before the RMU/RESTORE/ONLY_ROOT command will be ignored by the RMU/RECOVER command, which uses the transaction state of the database stored in the database root file to select which journaled transactions to recover from the After Image Journal (AIJ) files used by the RMU/RECOVER command. Important journaled updates to database parameters, such as the client sequence numbers maintained both in the database root (CLTSEQ) and the database system tables (RDB\$SEQUENCES), may be lost or made inconsistent.

The database should be verified both after the RMU/RESTORE/ONLY_ROOT command completes and after the RMU/RECOVER command completes. Please consult the documentation in the Oracle Rdb RMU Reference Manual for more information and examples of the options related to AIJ files and setting the database root transaction TSN and CSN values when the Rdb database root file (.rdb) is restored using the RMU/RESTORE/ONLY_ROOT command.

The TSN and CSN values set in the database root restored by the RMU/RESTORE/ONLY_ROOT command are displayed by an informational message if logging is enabled.

```
%RMU-I-SETRTTSNCSN, Setting Root Transaction Sequence TSN to 384,
  Commit Sequence CSN to 384
```

In this example, the /NOSET_TSN qualifier is used so that the TSN and CSN values of the restored root file are set to the values in the backup file used by the RMU/RESTORE/ONLY_ROOT command. As a result, the original journaled client sequence value of "21" is recovered by the RMU/RECOVER command executed after the RMU/RESTORE/ONLY_ROOT command.

```
$ sql$
SQL> attach 'file mf_personnel';
SQL> select RDB$SEQUENCE_NAME, RDB$NEXT_SEQUENCE_VALUE
cont> from rdb$sequences;
  RDB$SEQUENCE_NAME          RDB$NEXT_SEQUENCE_VALUE
  S                          21
1 row selected
exit;
$ delete mf_personnel.rdb;*
$ rmu/restore/only_root/log/NOSET_TSN mf_personnel
%RMU-I-AIJRSTBEG, restoring after-image journal "state" information
%RMU-I-AIJRSTJRN, restoring journal "AIJ1" information
%RMU-I-AIJRSTSEQ, journal sequence number is "0"
%RMU-I-AIJRSTSUC, journal "AIJ1" successfully restored from file
  "DEVICE:[DIRECTORY]AIJ_ONE.AIJ;1"
```

Oracle® Rdb for OpenVMS

```
%RMU-I-AIJRSTJRN, restoring journal "AIJ2" information
%RMU-I-AIJRSTNMD, journal has not yet been modified
%RMU-I-AIJRSTSUC, journal "AIJ2" successfully restored from file
"DEVICE:[DIRECTORY]AIJ_TWO.AIJ;1"
%RMU-I-AIJRSTEND, after-image journal "state" restoration complete
%RMU-I-SETRTTSNCSN, Setting Root Transaction Sequence TSN to 384,
Commit Sequence CSN to 384
%RMU-I-AIJISON, after-image journaling has been enabled
%RMU-W-DOFULLBCK, full database backup should be done to ensure
future recovery
%RMU-I-AIJRECEND, after-image journal "state" recovery complete
$
$ sql$
SQL> attach 'file mf_personnel';
SQL> select RDB$SEQUENCE_NAME, RDB$NEXT_SEQUENCE_VALUE
cont> from rdb$sequences;
RDB$SEQUENCE_NAME                RDB$NEXT_SEQUENCE_VALUE
S                                  1
1 row selected
exit;
$ rmu/recover/out=recov.sav  AIJ_ONE.aij,AIJ_TWO.aij
$
$ sql$
SQL> attach 'file mf_personnel';
SQL> select RDB$SEQUENCE_NAME, RDB$NEXT_SEQUENCE_VALUE
cont> from rdb$sequences;
RDB$SEQUENCE_NAME                RDB$NEXT_SEQUENCE_VALUE
S                                  21
1 row selected
exit;
$
```

14.1.12 Oracle Rdb Position on NFS Devices

This release note describes the supported usage of the NFS (Network File System) mounted devices by the Oracle Rdb product. NFS devices appear in most regards as local mounted file systems but do not allow the same level of sharing as provided by local OpenVMS devices. In addition, these files reside on a non-OpenVMS system (for instance a Linux or Windows system) and are therefore outside any scheme used by Rdb to lock buffers and pages of the database.

Active System Files

When Rdb is actively using database files, these files require specific sharing and locking to guarantee database integrity and recovery. Therefore, because of the limitations of the NFS mounted devices, active files such as the database root (.rdb), storage areas (.rda), snapshot files (.snp), row cache work file (.rdc), after image journal files (.aij), and before image recovery journal (.ruj) must not reside on an NFS mounted device.

Archived Data Files

Files that are not part of the active system may be stored on NFS mounted devices. For example, RMU /BACKUP /AFTER_JOURNAL can be used to archive an after image journal to a target on an NFS device. Similarly, RMU /BACKUP can perform a full or incremental backup to an Rdb backup file (.rbf) on an NFS device and RMU /RESTORE can use that NFS mounted source for database recovery, along with archived after image files from an NFS device processed by RMU /RECOVER.

Other Miscellaneous Files

Other files that might be used by an Rdb installation include options files, application procedures and sources, backup journals, record definitions files (.rrd), unloaded database files (.unl), exported databases (.rbr), log files, and so on. These sequential files may be stored on and referenced by RMU and SQL commands from an NFS mounted device.

Setting Up NFS

Complete instructions for setting up an NFS mounted device is beyond the scope of this release note and customers are directed to use system specific documentation for the server platform and for HPE OpenVMS systems. However, during testing with Oracle Rdb we noted the need for the following qualifiers for the TCPIP MOUNT command.

- Use /ADF=CREATE. This ensures that attributes (such as block size and record length) are preserved on the server.
- Use /STRUCTURE=5. This will emulate an ODS-5 device and therefore allow the most complete OpenVMS Files-11 On-Disk Structure emulation.
- Use /TRANSPORT=UDP. For example,

```
$ tcpip mount dnfs1:/host="test.company.com"/path="/scratch"
    /stru=5/serve=unix/adf/vers=2/tran=udp
```

Read Performance Issues

In versions of Oracle Rdb prior to Rdb V7.3.1.2, a significant performance issue exists when reading sequential files from NFS mounted devices. Oracle Rdb uses the RMS read-ahead (RAH) attribute to improve sequential reads but this has an adverse effect when referencing an NFS device. The latest release of Oracle Rdb works around this issue by disabling the use of read-ahead when referencing an NFS device and would be the preferred version when using NFS devices.

Disclaimer

This information is provided to answer customer questions and should not be read as an endorsement or guarantee for NFS systems. Oracle expects configuration, functional testing, performance testing, security and integrity of the NFS data to be performed by our customers.

14.1.13 RDM\$BIND_STAREA_EMERGENCY_DIR Logical Name

Bugs 19545970 and 3682207

RDM\$BIND_STAREA_EMERGENCY_DIR is a HOT STANDBY logical name that can be utilized when replicating the creation of a new storage area from a master database to its standby database.

RDM\$BIND_STAREA_EMERGENCY_DIR provides an alternate device and/or directory specification for the standby that can replace all or part of the master's file specification. Without the logical, the device and directory of the new storage area issued from the master must exist and match exactly on the standby. For example, on the master database we want to create a new starea, \$1SDGA11:[RDB_RANDOM.FOO]A1.RDA. We would issue the following command:

Oracle® Rdb for OpenVMS

```
SQL> alter database file rdb_random$db
      add storage area a1 filename $1$DGA11:[RDB_RANDOM.FOO]A1.RDA;
```

If the standby did not have a device called \$1\$DGA11, the replication would fail and the AIJ Log Roll-Forward Server (LRS) logfile would log the failure.

```
3-SEP-2014 16:22:26.94 - Replicating master FILID 19
3-SEP-2014 16:22:26.94 - Attempting to create starea
"$1$DGA11:[RDB_RANDOM.FOO]A1.RDA;1" ALQ=2808
3-SEP-2014 16:22:26.95 - Unable to create storage area. STATUS: 00DDA89C
3-SEP-2014 16:22:26.95 - No emergency directory defined
3-SEP-2014 16:22:26.95 - Failure reason: LRSSRV$CREATE_AREA_CALLBACK - Could
not create storage area
```

Suppose the target disk on the standby was \$1\$DGA109 and we defined the logical RDM\$BIND_STAREA_EMERGENCY_DIR to point to that, as in the following example.

```
$ define/sys RDM$BIND_STAREA_EMERGENCY_DIR "$1$DGA109:"
$ create/dir $1$DGA109:[RDB_RANDOM.FOO]
```

The replication operation would succeed and the LRS logfile would show:

```
3-SEP-2014 15:42:45.65 - Attempting to create starea
"$1$DGA11:[RDB_RANDOM.FOO]A1.RDA;1" ALQ=2808
3-SEP-2014 15:42:45.67 - Unable to create storage area. STATUS: 00DDA89C
3-SEP-2014 15:42:45.67 - Using emergency area "$1$DGA109:[RDB_RANDOM.FOO]A1.RDA"
3-SEP-2014 15:42:45.67 - Attempting to create starea
"$1$DGA109:[RDB_RANDOM.FOO]A1.RDA" ALQ=2808
3-SEP-2014 15:42:45.68 - Starea creation successful
3-SEP-2014 15:42:45.70 - Attempting to create starea
"$1$DGA11:[RDB_RANDOM.FOO]A1.SNP;1" ALQ=404
3-SEP-2014 15:42:45.70 - Unable to create storage area. STATUS: 00DDA89C
3-SEP-2014 15:42:45.70 - Using emergency area "$1$DGA109:[RDB_RANDOM.FOO]A1.SNP"
3-SEP-2014 15:42:45.70 - Attempting to create starea
"$1$DGA109:[RDB_RANDOM.FOO]A1.SNP" ALQ=404
3-SEP-2014 15:42:45.71 - Starea creation successful
```

The RDM\$BIND_STAREA_EMERGENCY_DIR logical must:

- Exist on the standby system prior to the create storage area operation.
- Be defined in the LNM\$SYSTEM_TABLE table.
- Be a valid file specification.

All standby databases on the node where the logical is defined share its use.

This logical was added back in Oracle Rdb Release 7.0.2 but the documentation of the logical was omitted.

14.1.14 RDMS-F-FULLAIJBKUP, Partially-Jounaled Changes Made

Bug 7669735

The Oracle Rdb and Oracle CODASYL DBMS Guide to Hot Standby Databases states: "You can stop replication operations by explicitly entering the Replicate After_Journal Stop command on either the standby

or master database nodes. Stopping replication on either database terminates replication on both databases."

Although the RMU/REPLICATE AFTER_JOURNAL STOP command may be issued against either Master or Standby to shut down replication, we have determined that there is at least one scenario where the choice is important relating to restarting replication in the future.

If you do the following, the operation will fail with a 'FULLAIJBKUP' error when starting the Master.

1. Stop replication on the Standby.
2. Set the old standby to be the new Master.
3. Set the old Master to be the new Standby.
4. Attempt to restart replication.

This is expected behavior. If the Standby is stopped prior to the Master, Oracle Rdb cannot determine if there has been any network traffic from the Master between the time that the Standby and Master shut down. Since any such information would be lost and may lead to data inconsistencies, replication will not be started.

The workaround for this scenario would be to stop replication on the Master, not the Standby. Consider the following two examples (assuming that replication is currently active):

Example 1: Initially stopping Replication on the Standby.

\$! Stopping Replication on the Standby:

```
$ RMU/REPLICATE AFTER STOP/WAIT/LOG STANDBY$DB:STANDBY_PERSONNEL
%RMU-I-HOTSTOPWAIT, stopping database replication, please wait
%RMU-I-LOGMODSTR, stopped master database AIJ Log Replication Server
```

\$! Start Replication of the Standby db (which was previously the Master)

```
$ RMU/REPLICATE AFTER_JOURNAL START MASTER:MF_PERSONNEL.RDB -
  /CHECKPOINT=10 -
  /LOG -
  /WAIT -
  /BUFFERS=30 -
  /GAP_TIMEOUT=5 -
  /GOVERNOR=DISABLED -
  /MASTER_ROOT=STANDBY$DB:STANDBY_PERSONNEL.RDB -
  /ONLINE
%RMU-I-LOGMODSTR, started standby database AIJ Log Replication Server
```

\$! Start Replication on the Master db (which was previously the Standby)

```
$ RMU/REPLICATE AFTER_JOURNAL START STANDBY$DB:STANDBY_PERSONNEL.RDB -
  /CHECKPOINT=100 -
  /LOG -
  /WAIT -
  /CONNECT_TIMEOUT=5 -
  /STANDBY_ROOT=MASTER:MF_PERSONNEL.RDB -
  /SYNCHRONIZATION=COLD -
  /QUIET_POINT -
  /TRANSPORT=TCPIP
%RMU-I-LOGMODSTR, started AIJ Log Server
%RDMS-F-CANTSTARTLCS, error starting AIJ Log Catch-Up Server process
-RDMS-F-FULLAIJBKUP, partially-journalled changes made; database may not be
recoverable
%RMU-F-FATALRDB, Fatal error while accessing Oracle Rdb.
```

Oracle® Rdb for OpenVMS

%RMU-F-FTL_RMU, Fatal error for RMU operation at 4-AUG-2014 14:19:17.78

Example 2: Initially stopping Replication on the Master.

\$! Stopping Replication on the Master:

```
$ RMU/REPLICATE AFTER STOP/WAIT/LOG MASTER$DB:MF_PERSONNEL.RDB
%RMU-I-HOTSTOPWAIT, stopping database replication, please wait
%RMU-I-LOGMODSTR, stopped master database AIJ Log Replication Server
```

\$! Start Replication of the Standby db (which was previously the Master)

```
$ RMU/REPLICATE AFTER_JOURNAL START MASTER:MF_PERSONNEL.RDB -
  /CHECKPOINT=10 -
  /LOG -
  /WAIT -
  /BUFFERS=30 -
  /GAP_TIMEOUT=5 -
  /GOVERNOR=DISABLED -
  /MASTER_ROOT=STANDBY$DB:STANDBY_PERSONNEL.RDB -
  /ONLINE
%RMU-I-LOGMODSTR, started standby database AIJ Log Replication Server
```

\$! Start Replication on the Master db (which was previously the Standby)

```
$ RMU/REPLICATE AFTER_JOURNAL START STANDBY$DB:STANDBY_PERSONNEL.RDB -
  /CHECKPOINT=100 -
  /LOG -
  /WAIT -
  /CONNECT_TIMEOUT=5 -
  /STANDBY_ROOT=MASTER:MF_PERSONNEL.RDB -
  /SYNCHRONIZATION=COLD -
  /QUIET_POINT -
  /TRANSPORT=TCPIP
%RMU-I-LOGMODSTR, started AIJ Log Server
%RMU-I-LOGMODSTR, started master database AIJ Log Replication Server
```

The SYS\$HELP:RMU_MSG*.DOC has more information about the FULLAIJBKUP error:

FULLAIJBKUP, partially-journalled changes made; database may not be recoverable

Explanation: Partially journalled changes have been made to the database. This may result in the database being unrecoverable in the event of database failure; that is, it may be impossible to roll-forward the after-image journals, due to a transaction mis-match or attempts to modify objects that were not journalled. This condition typically occurs as a result of replicating database changes using the Hot Standby feature.

User Action: IMMEDIATELY perform a full (not by-sequence) quiet-point AIJ backup to clear the AIJ journals, followed immediately by a full (no-quiet-point allowed) database backup.

14.1.15 Undocumented Hot Standby Logical Names

Bug 3264793

Table 14–3 Hot Standby Logical Names

Logical Name Description	Default Value	Minimum Value	Maximum Value
RDM\$BIND_ALS_LOG_REOPEN_SECS Defines the number of seconds after which the ALS output file will automatically be reopened.	0 seconds (will not be reopened automatically)	0 seconds	31449600 (1 year)
RDM\$BIND_ALS_LOG_REOPEN_SIZE Defines the number of blocks after which the ALS output file will automatically be reopened.	0 blocks (will not be reopened automatically)	0 blocks	Infinite
RDM\$BIND_HOT_ABS_SUSPEND_SHUTDOWN Defines whether or not the AIJ backup server (ABS) should be automatically suspended on graceful shutdown.	0	0	1
RDM\$BIND_HOT_CHECKPOINT Specifies the number of messages per server checkpoint interval. If specified, the first threshold to be exceeded (message count or elapsed time) will cause the checkpoint.	100	1	50000
RDM\$BIND_HOT_CHECKPOINT_INTERVAL Specifies a checkpoint interval, in minutes, to be used in addition to the /CHECKPOINT qualifier specified at Hot Standby startup. If specified, the first threshold to be exceeded (message count or elapsed time) will cause the LRS checkpoint.	0 minutes (don't use elapsed time)	0 minutes	10080 (7 days)
RDM\$BIND_HOT_IGNORE_NET_TIMEOUT Specifies whether or not to ignore network timeout parameters if the LRS process is still active.	0	0	1
RDM\$BIND_HOT_LOG_REOPEN_SECS Defines the number of seconds after which the AIJSERVER output file will automatically be reopened.	0 seconds (will not be reopened automatically)	0 seconds	604800 (1 week)
RDM\$BIND_HOT_LOG_REOPEN_SIZE Defines the number of blocks after which the AIJSERVER output file will automatically be reopened.	0 blocks (will not be reopened automatically)	0	Infinite
RDM\$BIND_HOT_NETWORK_ALT_NODE Defines the secondary network nodename to be used in the event of primary nodename network failure. This logical name allows you to specify an alternate routing pathway to the same standby database.	None		
RDM\$BIND_HOT_NETWORK_RETRY Specifies a network retry timeout interval.	120 seconds	0	1800 (30 minutes)
RDM\$BIND_LCS_AIJ_SCAN_IO_COUNT Defines the number of asynchronous I/O operations to be	64	1	128

performed simultaneously during LCS catch-up.			
RDM\$BIND_LCS_LOG_REOPEN_SECS Defines the number of seconds after which the LCS output file will automatically be reopened.	0 seconds (will not be reopened automatically)	0 seconds	31449600 (1 year)
RDM\$BIND_LCS_LOG_REOPEN_SIZE Defines the number of blocks after which the LCS output file will automatically be reopened.	0 blocks (will not be reopened automatically)	0 blocks	Infinite
RDM\$BIND_LCS_QUIET_TIMEOUT Defines the number of seconds to wait for the LCS process to obtain the standby database quiet-point.	600 seconds	0 seconds (wait indefinitely)	Infinite
RDM\$BIND_LCS_SYNC_COMMIT_MAX Defines the number of catch-up messages to synchronize with the standby database. A message may contain multiple transactions.	128 messages	32 messages	10000 messages
RDM\$BIND_LRS_LOG_REOPEN_SECS Defines the number of seconds after which the LRS output file will automatically be reopened.	0 seconds (will not be reopened automatically)	0 seconds	31449600 (1 year)
RDM\$BIND_LRS_LOG_REOPEN_SIZE Defines the number of blocks after which the LRS output file will automatically be reopened.	0 blocks (will not be reopened automatically)	0 blocks	Infinite
RDM\$BIND_LRS_QUIET_TIMEOUT Defines the number of seconds to wait for the LRS process to obtain the standby database quiet-point.	600	0 seconds (wait indefinitely)	Infinite
RDM\$BIND_STAREA_EMERGENCY_DIR Defines an alternate device and directory for the creation of storage areas on the standby database. The logical must be defined in the LNM\$SYSTEM_TABLE table and it is shared by all standby databases on that node.			

14.1.16 Missing Documentation for the TRANSACTION_TYPE Keyword for GET DIAGNOSTICS

Prior versions of the SQL Reference Manual omitted the description of the TRANSACTION_TYPE keyword for GET DIAGNOSTICS.

TRANSACTION_TYPE returns the type of transaction being executed. The result will be one of the following strings: 'BATCH UPDATE', 'READ ONLY', 'READ WRITE', or 'NONE'.

The result data type is CHAR (31).

Examples

Within a compound statement, you can use GET DIAGNOSTICS to retrieve information about the query state and its environment. In this example, we use the GET DIAGNOSTICS keywords TRANSACTION_TYPE and ROW_COUNT.

Example 14-1 Example: Using TRANSACTION_TYPE to control actions of a procedure

```

SQL> attach 'file MF_PERSONNEL';
SQL>
SQL> -- Sample procedure to use GET DIAGNOSTICS
SQL>
SQL> declare :rc integer;
SQL> declare :txn_type char(31);
SQL>
SQL> begin
cont>     set :rc = 0;
cont>     get diagnostics :txn_type = transaction_type;
cont>     trace '' || :txn_type || '';
cont>     case :txn_type
cont>         when 'BATCH UPDATE' then
cont>             begin
cont>                 -- do nothing
cont>             end;
cont>         when 'READ ONLY' then
cont>             rollback;
cont>         when 'READ WRITE' then
cont>             delete from employees;
cont>             get diagnostics :rc = row_count;
cont>             trace 'Rows deleted = ', :rc;
cont>         when 'NONE' then
cont>             begin
cont>                 -- no transaction so start one
cont>                 set transaction read only;
cont>             end;
cont>     end case;
cont> end;
SQL>
SQL> print :txn_type, :rc;
  TXN_TYPE                RC
  NONE                    0
SQL>
SQL> rollback;

```

14.1.17 Clarification on Using the RMU/UNLOAD TRIM=TRAILING Option

The following example shows that unexpected results may occur with the RMU/UNLOAD command Trim option when spaces are unloaded from an Oracle Rdb database.

Create a table in an Rdb database with two character columns and insert spaces and other character data into the table fields.

```

SQL> create database filename testdb;
create table tabl(col1 char(2), col2 char(2));
insert into tabl values (' ', ' ');
insert into tabl values ('AB', ' ');
insert into tabl values (' ', 'CD');
insert into tabl values ('A ', 'C ');
commit;

```

Unload the character field data from the table, specifying the Trim=trailing option to eliminate trailing spaces but do not specify prefix or suffix delimiter values.

Oracle® Rdb for OpenVMS

```
$ rmu/unload/record=(file=tabl, -  
  format=delimited_text, prefix="", suffix="", separator="|", -  
  null="NULL", trim=trailing) testdb tabl tabl
```

The trailing spaces are eliminated from the unload file since the Trim=trailing option was used.

```
$ ty tabl.unl  
|  
AB|  
|CD  
A|C
```

Now load the unloaded data back into the database table from the unload file.

```
$ rmu/load/log/record=(file=tabl, -  
  format=delimited_text, prefix="", suffix="", separator="|", -  
  null="NULL") testdb tabl tabl
```

This is the result:

```
SQL> att 'f testdb';  
SQL> select '>' || col1 || '<  ', '>' || col2 || '<' from tabl;  
  
> <      > <  
>AB<      > <  
> <      >CD<  
>A <      >C <  
> <      NULL  
>AB<      NULL  
> <      >CD<  
>A <      >C <  
8 rows selected
```

The first and second row, which originally contained two spaces in COL2, are now set to NULL. This happens because of the use of the option Trim=trailing in the RMU/UNLOAD command.

Because neither prefix nor suffix characters are specified in the RMU/UNLOAD command, it cannot be determined whether values existed in the character fields which only contained spaces or if these fields were flagged as NULL fields in the database.

The trailing column is set to NULL as described by the Oracle Rdb RMU Reference Manual which states:

"If the final column or columns of a record are to be set to NULL, you only have to specify data for the column up to the last non-null column. See the Examples section for an example of each of these methods of storing the NULL value."

Therefore, a trailing empty field will be null. Inner columns will be null if set to the string specified by the NULL option.

To avoid this result, you could eliminate the TRIM option in the RMU/UNLOAD command, or if you need the TRIM option then you can avoid this result by specifying a character value for the PREFIX and SUFFIX separator options for both the RMU/LOAD and the RMU/UNLOAD commands as in the following example.

```
$ rmu/unload/record=(file=tabl, -  
  format=delimited_text, prefix="**", suffix="**", separator="|", -  
  null="NULL", trim=trailing) testdb tabl tabl
```

```

$
$ ty tabl.unl
**|**
*AB*|**
**|*CD*
*A*|*C*
$
$ rmu/load/log/record=(file=tabl, -
  format=delimited_text, prefix="*", suffix="*", separator="|", -
  null="NULL") testdb tabl tabl

```

14.1.18 Corrections to the EDIT STRING Documentation

Bugs 17365476 and 17365597

- The SQL Reference Manual, Volume 1, incorrectly stated that fields following a quoted literal would have leading zeros trimmed if the literal ended with a space. This was incorrect. The trimming only takes place after a space formatting character (B).

This is the corrected text:

Oracle Rdb automatically trims leading zeros from the first numeric field in the output, and any numeric field following a space formatting character (B). The year (Y) and fractional seconds (*) format fields are never trimmed of leading zeros.

- To have SQL represent an OpenVMS date format without removing the leading zero from the Hour field, use the literal string for space rather than the space formatting character (B).

```
edit string 'YYYY-NN-DD" "RR:PP:QQ.**'
```

rather than

```
edit string 'YYYY-NN-DDBRR:PP:QQ.**'
```

- The formatting string ** represents the 100ths of a second field. Prior versions using a narrow field * would erroneously truncate the leading digits. This is corrected in this release, as the trailing digit is truncated.

14.1.19 Changes and Improvements to the Rdb Optimizer and Query Compiler

This release of Oracle Rdb introduces several new capabilities within the query compiler and the query optimizer. These changes fall generally under the title *query rewrite*, and allow the query compiler to present a simplified query for optimization and execution.

- CAST function elimination

In most cases, CAST actions must be executed at runtime to convert from the source data type to that specified by the CAST function. However, in some cases, the Rdb query compiler can eliminate or replace the CAST function with a literal value during query compile. This saves CPU time as the action is performed just once rather than once per row processed.

This replacement includes the following:

- ◆ When CAST of DATE (ANSI), DATE (VMS) or TIMESTAMP data types is performed to a

compatible type of DATE or TIMESTAMP, then in many cases the CAST operator is not required.

- ◆ CAST of string literals to DATE (ANSI), DATE (VMS), TIME, TIMESTAMP and INTERVAL can be processed at compile time. For example, CAST('2013-1-1' AS DATE ANSI) is implicitly converted to a DATE literal DATE'2013-1-1'.
- ◆ CAST of small integer values is now done by the compiler. For example, CAST(1 AS SMALLINT) can be performed at compile time.
- ◆ CAST of fixed length (CHAR) literal strings to varying length strings (VARCHAR) is now processed by the compiler if the character set is the same and the target VARCHAR is long enough to hold the source string, as seen in the following example:

```
CAST('TABLE' AS VARCHAR(31))
```

- Constant Folding

Simple arithmetic expressions involving integer or floating point literals are evaluated by the query compiler. The overall effect is smaller executable code and some reduced CPU time for queries. FLOAT, REAL, and DOUBLE PRECISION values are combined to produce DOUBLE PRECISION results. Integer literals (with no fractional component) are combined to produce BIGINT results. The side effect is that some expressions may now return DOUBLE PRECISION or BIGINT results where in prior versions they produced smaller precision results. This should not affect applications which fetch values into different data types as Oracle Rdb will perform an implicit conversion.

This optimization includes the following:

- ◆ Addition (+)
- ◆ Subtraction (-)
- ◆ Multiplication (*)
- ◆ Division (/)

Note that division is not performed at compile time if the divisor is a literal zero (0).

Operations which are coded to explicitly divide by zero are probably expected to produce an error at runtime. Although using the SQL SIGNAL statement is now preferred, this technique has been used to terminate procedures when an incorrect input is encountered.

- Algebraic Rules

Additive identity (zero) can be added to an expression without changing the value. The query compiler will eliminate the literal zero (0) from the expression.

Multiply by zero will result in zero if the other operand is a not nullable expression. In this case, the expression will be replaced by zero.

Multiplicative identity (one) can be multiplied by an expression without changing the value. The query compiler will eliminate the literal one (1) from the expression.

The side effect is that some expressions may now return slightly different data types because the literal is no longer considered as part of the data type computation.

- Simple Predicate Elimination

When predicates include comparison of simple expressions, then the query compiler will attempt to eliminate them from the query predicate. For example, WHERE ('A' = 'A') will be replaced by TRUE, WHERE (2 <> 2) will be replaced with FALSE, and so on.

- Not Nullable Aware

The query compiler is now aware of which columns have a NOT NULL NOT DEFERRABLE constraint enabled. Additionally, this attribute is also implied from any PRIMARY KEY NOT DEFERRABLE constraints.

Using this knowledge, the query compiler can reduce (prune) the query expression. This list defines the ways in which this can occur:

- ◆ When IS NULL is applied to a not nullable column or expression, then this predicate is replaced with FALSE.

- ◆ When IS NOT NULL is applied to a not nullable column or expression, then this predicate is replaced with TRUE.

The side effect is that constraints for a table are now loaded for SELECT statements.

This optimization can be disabled using the SET FLAGS statement, or the RDMS\$SET_FLAGS logical name with the value NOREWRITE(IS_NULL). The default is REWRITE(IS_NULL).

- Replace comparisons with NULL

Queries that erroneously compare value expressions with NULL will now be replaced with a simplified UNKNOWN value. For example, a query that uses WHERE EMPLOYEE_ID = NULL will never find matching rows, because the results of the comparison (equals, not equals, greater than, less than, and so on) are always UNKNOWN.

This optimization can be disabled using the SET FLAGS statement, or the RDMS\$SET_FLAGS logical name with the value NOREWRITE(UNKNOWN). The default is REWRITE(UNKNOWN).

- Predicate Pruning

The AND, OR and NOT operators can be simplified if the logical expressions have been reduced to TRUE, FALSE or UNKNOWN expressions. Depending on the operation, the Rdb query compiler might be able to eliminate the Boolean operator and part of the expression.

This optimization can be disabled using the SET FLAGS statement, or the RDMS\$SET_FLAGS logical name with the value NOREWRITE(BOOLEANS). The default is REWRITE(BOOLEANS).

- CASE Expression Pruning

The prior transformation will also be applied to the Boolean WHEN expressions of a conditional expression (CASE, DECODE, NULLIF, COALESCE, NVL, NVL2, SIGN, ABS, and so on).

In some cases, the resulting conditional expression might resolve to an equivalent conditional expression with fewer branches (some WHEN ... THEN clauses being eliminated) or a simple expression with no conditional expression (all WHEN ... THEN clauses are eliminated).

- IN Operator Simplification

The IN operator using a subquery looks similar to the EXISTS boolean expression but it differs in its handling of NULL values. If the query compiler knows that neither source field nor the value set contain NULL, then the EXISTS expression can replace the IN operator. The EXISTS expression generates a better query solution in almost all cases.

This optimization can be disabled using the SET FLAGS statement, or the RDMS\$SET_FLAGS logical name with the value NOREWRITE(IN_CLAUSE). The default is REWRITE(IN_CLAUSE).

In most cases, the results of these optimizations will be transparent to the application. However, database administrators that use SET FLAGS 'STRATEGY,DETAIL' will notice new notations in the displayed strategy.

The following examples show the types of likely results.

In this example, the logical expression (1 = 2) is replaced with FALSE, the logical expression (1 = 1) is replaced with TRUE and the predicate is reduced to just the IS NULL (aka MISSING) check.

```
SQL> select last_name
cont> from employees
cont> where ((1 = 1) and employee_id is null)
cont>         or
cont>         ((1 = 2) and employee_id = '00164');
Tables:
  0 = EMPLOYEES
Conjunct: MISSING (0.EMPLOYEE_ID)
Get      Retrieval sequentially of relation 0:EMPLOYEES
0 rows selected
```

If there existed a NOT NULL NOT DEFERRABLE constraint on the EMPLOYEE_ID column, the expression can be further reduced because the NOT NULL constraint means the IS NULL test is always FALSE.

```
SQL> alter table EMPLOYEES
cont>     alter column EMPLOYEE_ID
cont>         constraint NN_EMPLOYEE_ID
cont>             NOT NULL
cont>             NOT DEFERRABLE
cont> ;
SQL>
SQL> select last_name
cont> from employees
cont> where ((1 = 1) and employee_id is null)
cont>         or
cont>         ((1 = 2) and employee_id = '00164');
Tables:
  0 = EMPLOYEES
Conjunct: FALSE
Get      Retrieval sequentially of relation 0:EMPLOYEES
0 rows selected
SQL>
```

REWRITE Flag

The SET FLAGS statement and the RDMS\$SET_FLAGS logical name can be used to enable or disable some of these rewrite actions. This flag primarily exists for Oracle to test the behavior of the query rewrite changes. It can be used by programmers to revert to pre-V7.3 behavior.

REWRITE enables each rewrite setting and NOREWRITE disables them. Additionally, keywords can be added to REWRITE and NOREWRITE to disable selective rewrite actions.

The following new keywords are added for this release of Oracle Rdb.

- BOOLEANS
- IN_CLAUSE
- IS_NULL
- UNKNOWN

14.1.20 Required Privileges for AUTHORIZATION Clause of CREATE MODULE

The following usage note is missing from the SQL Reference Manual, under the CREATE MODULE Statement.

- When the AUTHORIZATION clause is used, the definer of the module is granting his/her own privileges to the specified username so that tables, columns, sequences, procedures and functions are accessed as though accessed by the definer.
The AUTHORIZATION is expected to be the session user, or an OpenVMS rights identifier granted to that user (when SECURITY CHECKING IS EXTERNAL). If the session is run with one of the following OpenVMS privileges, then any user or rights identifier can be referenced: SYSPRV, BYPASS or IMPERSONATE.

Note

The OpenVMS IMPERSONATE privilege can be used to override the checking for Oracle Rdb Release 7.2.5.1 and later versions.

14.1.21 Sorting Capabilities in Oracle Rdb

Oracle Rdb supports both the traditional OpenVMS SORT32 facility as well as a simplified internal sort facility called QSORT.

QSORT

Use of QSORT preempts use of all other sorting algorithms. The QSORT algorithm is used if sorting is being done on a single key and if only a small amount of data is involved. The reason for this is that the other sorting algorithms, while using more efficient methods, have a certain amount of overhead associated with setting them up and with being general purpose routines.

QSORT is used by default if:

- There is a single sort key.
- The number of rows to be sorted is 5000 or fewer.
- The sort key is not floating point (REAL, FLOAT, or DOUBLE PRECISION).

How to Alter QSORT Usage

To change the usage of QSORT to evaluate behavior with other parameters, define a new row limit as follows:

```
$ DEFINE RDMS$BIND_MAX_QSORT_COUNT m
```

The default value is 5000 rows.

Note

Defining the logical RDMS\$BIND_MAX_QSORT_COUNT as 63 will return QSORT behavior to that used by prior releases of Oracle Rdb V7.2.

To disable QSORT because of either anomalous or undesirable performance, the user can define the following logical to the value zero, in which case the VMS SORT interface is always used.

```
$ DEFINE RDMS$BIND_MAX_QSORT_COUNT 0
```

14.1.22 RMU /VERIFY Process Quotas and Limits Clarification

When using the RMU/VERIFY command, a process requires a minimum of the following quotas:

- FILLM and CHANNELCNT at least 25 more than the total number of database storage areas, snapshot storage areas, and after image journals.
- Large enough BYTLM, page file quota and working set to open all of the database storage areas, snapshot storage areas, and after image journals.

14.1.23 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 defines 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 applies 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 need to be closed and reopened for a new value of the logical to become effective.

14.1.24 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 14-4](#).

Table 14-4 Server Process Priority Logical Names

Logical Name	Use
RDM\$BIND_ABS_PRIORITY	Base Priority for the ABS Server process
RDM\$BIND_ALS_PRIORITY	Base Priority for the ALS Server process
RDM\$BIND_DBR_PRIORITY	Base Priority for the DBR Server process
RDM\$BIND_LCS_PRIORITY	Base Priority for the LCS Server process
RDM\$BIND_LRS_PRIORITY	Base Priority for the LRS Server process
RDM\$BIND_RCS_PRIORITY	Base Priority for the RCS Server process

The RDMAIJSERVER account for Hot Standby 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.

14.1.25 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.

14.1.26 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 14–5 shows the TRANS_TPB table.

Table 14–5 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 14–6 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 14–6 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)	

14.1.27 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 14–7 shows the DATABASE table.

Table 14–7 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 14–8 shows the REQUEST_ACTUAL table.

Table 14–8 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 14–9 shows the TRANSACTION table.

Table 14–9 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 14–10 shows the REQUEST_BLR table.

Table 14–10 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

Oracle® Rdb for OpenVMS

REQUEST_TYPE	INTEGER	
BLR	VARCHAR(127)	
BLR_STR_ID	INTEGER	STR_ID_DOMAIN

14.2 RDO, RDBPRE and RDB\$INTERPRET Features

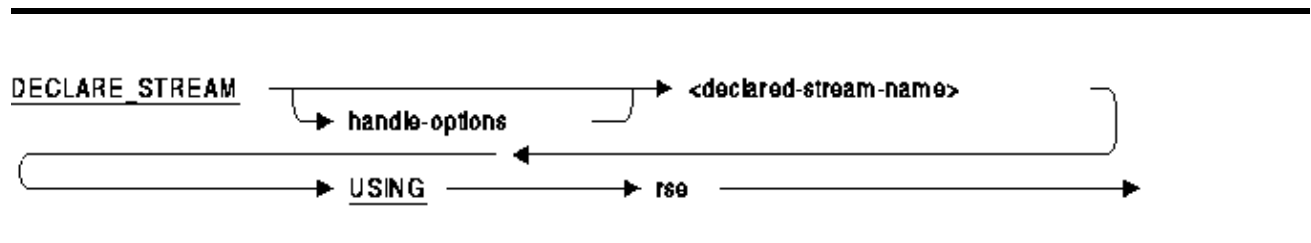
14.2.1 New Request Options for RDO, RDBPRE and RDB\$INTERPRET

For release 7.0.1 of Oracle Rdb, two new keywords were 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.

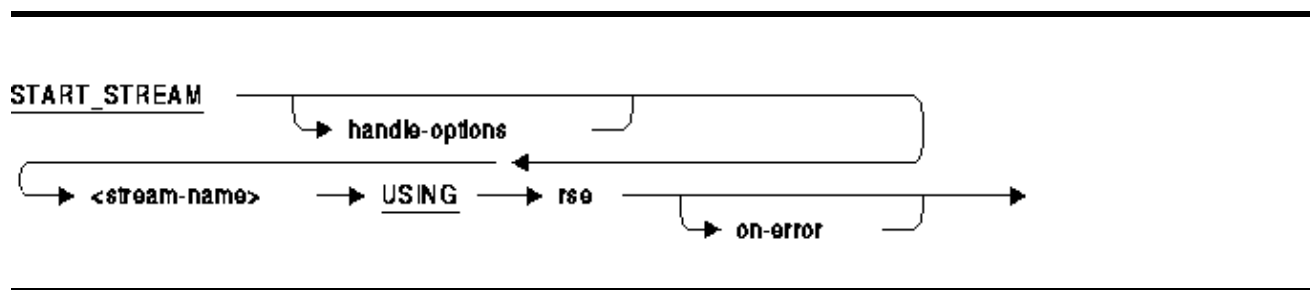
In prior releases, the handle-options could not be specified in interactive RDO or RDB\$INTERPRET. This has changed in Rdb but these allowed options will be limited to MODIFY and PROTECTED keywords. For RDBPRE, all options listed will be supported. These option names were chosen to be existing keywords to avoid adding any new keywords to the RDO language.

The altered statements are shown below.

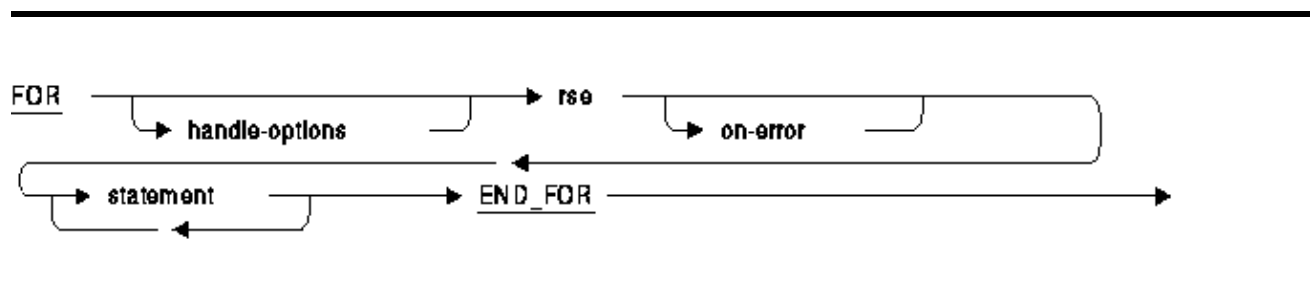
DECLARE_STREAM Format



START_STREAM Format

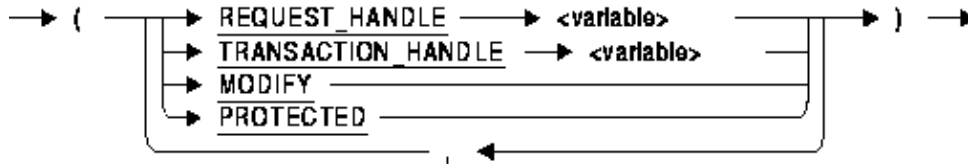


FOR Format



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. P> 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 ' +
              '!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.

The problem was corrected in Oracle Rdb Release 7.0.1.

14.2.2 New Language Features for RDO and Rdb Precompiler

The following new language enhancements have been made to RDO, the Rdb Precompiler (RDBPRE), and the RDO Interpreter (RDB\$INTERPRET).

- LIKE operator

```
--> <value_expr> LIKE <value_expr> -->
```

The rse WITH clause can now specify a LIKE relational operator, which is similar in action to the MATCHING operator. The LIKE operator returns TRUE if the second expression pattern matches the first value expression. LIKE is case sensitive. LIKE uses these special characters:

- ◆ % Matches any string
- ◆ _ Matches any character
- ◆ \ an escape character. Use \\ to represent a single \, \% to represent a literal "%", and _ to represent a literal "_".

This example is looking for any names starting with one character followed by an apostrophe.

```

RDO> for e in employees
cont>   with e.last_name like '_%'
cont>   print e.last_name
cont> end_for
LAST_NAME
D'Amico
O'Sullivan
RDO>

```

- FIRST VIA ... FROM sub-query expression

RDO includes a FIRST ... FROM sub-query expression. It returns the value from the matching row. However, if no rows are found, then the query will be aborted with a returned exception.

The following example wishes to list each relation and its associated storage map (if it exists), and shows the reported RDB-E-FROM_NO_MATCH error.

Oracle® Rdb for OpenVMS

```
RDO> for r in rdb$relations
cont>   with r.rdb$system_flag = 0
cont>   sorted by r.rdb$relation_name
cont>   print r.rdb$relation_name,
cont>     first sm.rdb$map_name from sm in rdb$storage_maps with
cont>       sm.rdb$relation_name = r.rdb$relation_name
cont> end_for
RDB$RELATION_NAME          SM.RDB$MAP_NAME
CANDIDATES                 CANDIDATES_MAP
%RDB-E-FROM_NO_MATCH, no record matched the RSE in a "from" expression
RDO>
```

RDO now supports an alternative to the FIRST ... FROM sub-query expression which modifies the behavior when no matching rows were selected by the sub-query. Adding the VIA keyword requests that a MISSING value be returned in such cases, and the query is no longer aborted.

```
RDO> for r in rdb$relations
cont>   with r.rdb$system_flag = 0
cont>   sorted by r.rdb$relation_name
cont>   print r.rdb$relation_name,
cont>     first via sm.rdb$map_name from sm in rdb$storage_maps with
cont>       sm.rdb$relation_name = r.rdb$relation_name
cont> end_for
RDB$RELATION_NAME          SM.RDB$MAP_NAME
CANDIDATES                 CANDIDATES_MAP
COLLEGES                   COLLEGES_MAP
CURRENT_INFO
CURRENT_JOB
CURRENT_SALARY
DEGREES                     DEGREES_MAP
DEPARTMENTS                 DEPARTMENTS_MAP
EMPLOYEES                   EMPLOYEES_MAP
EMPS
JOBS                         JOBS_MAP
JOB_HISTORY                 JOB_HISTORY_MAP
RESUMES                     RESUMES_MAP
SALARY_HISTORY              SALARY_HISTORY_MAP
WORK_STATUS                 WORK_STATUS_MAP
RDO>
```

- New special functions: RDO\$CURRENT_USER, RDO\$SESSION_USER and RDO\$SYSTEM_USER

These functions return the user identification of the current, session and system users. They can appear in any place that a field (aka column) can be used. These functions simplify view and trigger definitions created through RDO.

Any view, computed by field, or trigger created by RDO but executed by a SQL session may return different values for each function. However, RDO sessions will typically return the same value from each function.

This query uses the RDO\$CURRENT_USER function to select the tables and views created by a user.

```
RDO> for r in rdb$relations
cont>   with r.rdb$relation_creator = rdo$current_user
cont>   print r.rdb$relation_name, r.rdb$created
cont> end_for
RDB$RELATION_NAME          RDB$CREATED
EMKP                       23-JUN-2014 12:53:26.28
PICK_HISTORY_REC           31-JUL-2015 15:11:47.46
```

TEST_TABLE	8-AUG-2014 08:21:25.96
CUSTOMER_REC	8-AUG-2014 08:17:16.34
TAB1	8-AUG-2014 08:17:17.88
TAB2	8-AUG-2014 08:17:17.88
JOB_HIST	15-AUG-2014 13:49:07.39
SAL_HIST	15-AUG-2014 13:49:07.39
EMP_NAMES	14-OCT-2014 21:18:49.03
CAND_NAMES	14-OCT-2014 21:18:49.03
ACTION_CODES	14-OCT-2014 21:18:49.23
.	
.	
.	

14.2.3 RDO Interface Now Supports Synonym References

Bug 20355063

This release of Oracle Rdb adds minimal support for synonyms to RDO and RDBPRE. In prior versions, a synonym to a table (or view) was not recognized by the RDO interfaces. For instance, an application built against table names which were subsequently renamed using SQL would no longer compile because the synonyms established by the `RENAME TABLE` or `ALTER TABLE ... RENAME TO` statements were not recognized by RDBPRE or RDO.

This support allows queries that reference table or view synonyms to be processed by the RDBPRE precompiler and RDO interactive utility. In addition, most `SHOW` commands in RDO will recognize a table or view synonym.

Data definition (DDL) commands, such as `DROP RELATION` or `DEFINE CONSTRAINT`, do not accept a synonym name as input. For such operations, Oracle recommends using the Interactive SQL interface.

This problem has been corrected in Oracle Rdb Release 7.3.2.0. Synonyms for tables and views created using any of the following statements are now recognized by RDO.

- `RENAME TABLE ...`
- `RENAME VIEW ...`
- `ALTER TABLE ... RENAME TO ...`
- `ALTER VIEW ... RENAME TO ...`
- `CREATE SYNONYMS ... FOR TABLE ...`
- `CREATE SYNONYMS ... FOR VIEW ...`

14.3 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

14.4 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 the Oracle Rdb Documentation web page:

<http://www.oracle.com/technetwork/products/rdb/documentation/index.html>

Customers should contact their Oracle representative to purchase printed documentation.

Chapter 15

Known Problems and Restrictions

This chapter describes problems and restrictions relating to Oracle Rdb and includes workarounds where appropriate.

15.1 Known Problems and Restrictions in All Interfaces

This section describes known problems and restrictions that affect all interfaces. This is not an exhaustive list. Check the Oracle Bug database to see a list of all open Rdb bugs and their current status.

15.1.1 The Format of the RMU/ANALYZE/BINARY_OUTPUT Files Can Change

Bug 19261529

The /BINARY_OUTPUT qualifier used with the Oracle Rdb RMU/ANALYZE command specifies a binary output *.UNL file and/or a record definition *.RRD file to be created by RMU/ANALYZE to save statistical data created while analysing an Oracle Rdb database and to then load the statistical data into an Oracle Rdb database table. This data can then be used by a user-written management application or procedure. See the Oracle Rdb RMU REFERENCE MANUAL for more information on the /BINARY_OUTPUT qualifier used with the RMU/ANALYZE command.

The format of the binary and record definition files created by the RMU/ANALYZE/BINARY_OUTPUT command can change as changes are made to the statistical data produced by RMU/ANALYZE. Creators and maintainers of user-written management applications or procedures dependent on the format of these files should be aware that Oracle Rdb reserves the right to add, remove or move fields in this record definition file for each major release of Rdb, which can require changes in these user-written management applications or procedures. The release notes for each major version of Oracle Rdb will document any changes in the format of these files. To make users aware of this possibility, the following warning comment has now been added to the record definition file.

```
DEFINE RECORD name
DESCRIPTION IS /* Oracle Rdb V7.3-130
                Oracle reserves the right to add, remove or move fields in
                this record definition file for each major release of Rdb */ .
```

The following example shows a database table named POS being created to hold RMU/ANALYZE/PLACEMENT statistics. Then an RMU/ANALYZE/PLACEMENT/BINARY_OUTPUT command is executed to save binary placement statistics, for the EMP_EMPLOYEE_ID index in the MF_PERSONNEL database, to the POS.UNL file and the record format for these statistics to the POS.RRD file. The RMU/LOAD command is then executed to load the binary placement statistic records from the POS.UNL file into the POS table using the record field format defined in the POS.RRD file.

```
$ sql
attach 'file mf_personnel';

create table pos (
    RMU$DATE          DATE VMS,
    RMU$INDEX_NAME    CHAR(32),
    RMU$RELATION_NAME CHAR(32),
    RMU$PARTITION_NAME CHAR(32),
    RMU$AREA_NAME     CHAR(32),
    RMU$LEVEL         INTEGER,
    RMU$FLAGS         INTEGER,
    RMU$COUNT        INTEGER,
```



```

RMU$DUPLICATE_COUNT      INTEGER,
RMU$DUPLICATE_MAP_COUNT  INTEGER,
RMU$KEY_COUNT            INTEGER,
RMU$DUPLICATE_KEY_COUNT  INTEGER,
RMU$DATA_COUNT           INTEGER,
RMU$DUPLICATE_DATA_COUNT INTEGER,
RMU$TOTAL_KEY_PATH       INTEGER,
RMU$TOTAL_PAGE_PATH      INTEGER,
RMU$TOTAL_BUFFER_PATH    INTEGER,
RMU$MAX_KEY_PATH         INTEGER,
RMU$MAX_PAGE_PATH        INTEGER,
RMU$MIN_BUF_PATH         INTEGER);

commit;
exit
$ rmu/analyze /placement -
  /binary_output=(file=pos.unl,record_definition=pos.rrd) -
  mf_personnel EMP_EMPLOYEE_ID
$ rmu/load/rms=(file=pos.rrd) mf_personnel POS pos.unl
%RMU-I-DATRECREAD, 1 data records read from input file.
%RMU-I-DATRECSTO, 1 data records stored 3-DEC-2014 10:58:04
$ type pos.rrd

DEFINE RECORD RMU$ANALYZE_PLACEMENT
DESCRIPTION IS /* Oracle Rdb V7.3-130
                Oracle reserves the right to add, remove or move fields in
                this record definition file for each major release of Rdb */ .

$

```

15.1.2 RMU /VERIFY /KEY_VALUES May Fail on Some Indices

In some cases, the RMU/VERIFY/KEY_VALUES functionality may be unable to perform key value verification failing with an %RDMS-F-OUTLINE_FAILED error. This happens when the generated SQL query and query outline can not be used to ensure index only retrieval from the specified index. This may occur with multi-segment HASHED (ORDERED or SCATTERED) indices, or multi-segment SORTED (and SORTED RANKED) indices with the REVERSE attribute.

The following example shows the reported error.

```

$ rmu/verify/nolog/index=T5_IDENT_IMAGE_NDX /key_values VERIFY_KEY_VALUES_DB
%RDMS-F-OUTLINE_FAILED, could not comply with mandatory query outline directives
%RMU-I-NOTREQVFY, not all requested verifications have been performed
$

```

This problem is a known limitation in this release of Oracle Rdb.

15.1.3 REPLACE Statement Fails With Primary Key Constraint Failure When Used on a View

The REPLACE statement does not show the correct semantics when used to insert into a view defined on a table with a PRIMARY KEY constraint.

The following example shows the problem.

```

SQL> create table SAMPLE
cont>     (ident integer
cont>     ,prod_name char(20)
cont>     ,primary key (ident) not deferrable
cont>     );
SQL>
SQL> create view SAMPLE_VIEW
cont>     (ident, prod_name)
cont>     as select * from SAMPLE
cont> ;
SQL>
SQL> insert into SAMPLE values (1, 'Ajax');
1 row inserted
SQL> insert into SAMPLE_VIEW values (2, 'Mr Clean');
1 row inserted
SQL>
SQL> replace into SAMPLE values (1, 'Borox');
1 row replaced
SQL> replace into SAMPLE_VIEW values (2, 'Mr. Clean');
%RDB-E-INTEG_FAIL, violation of constraint SAMPLE_PRIMARY_IDENT caused operation to fail
-RDB-F-ON_DB, on database RDB$DEFAULT_CONNECTION
SQL>
SQL> select * from SAMPLE order by ident;
      IDENT  PROD_NAME
        1    Borox
        2    Mr Clean
2 rows selected
SQL>

```

This will remain a restriction for this release. Only use REPLACE (or RMU/LOAD/REPLACE) on base tables.

15.1.4 Possible Incorrect Results When Using Partitioned Descending Indexes

Bug 6129797

In the current release of Oracle Rdb, it is possible for some queries using partitioned indexes with segments of mixed ascending and descending order to return incorrect results either on Alpha or I64 systems.

The following examples show two problems when using partitioned index with segments of mixed ascending and descending order:

```

create database file foo
  create storage area fooa
  create storage area foob;

create table mesa (id integer, m4 char (1), m5 integer);
create table rasa (id integer, r4 char (1), r5 integer);

insert into mesa (id, m4, m5) values (1, 'm', 1 );

insert into rasa (id, r4, r5) values (1, 'm', 1 );
insert into rasa (id, r4, r5) values (1, 'k', 1 );
insert into rasa (id, r4, r5) values (1, 'e', 1 );

```

Oracle® Rdb for OpenVMS

```
create index x4 on mesa (id asc , m4 asc) ;

! The following index contains ascending segments followed by descending
! segments and thus causes the query to return the wrong result.
!
! Note that the query works if both segments are either ascending or descending.
!
create index y4 on rasa (id asc , r4 desc)
  store using (id, r4)
  in foaa with limit of (1, 'g' )
  otherwise in foob ;
commit;

! Problem #1:
!
! the following query returns correctly 3 rows on Alpha but 1 row on IA64:

SQL> select m.id, m.m4, r.r4 from
  mesa m inner join rasa r on (m.id = r.id);
      1  m      m
      1  m      k
      1  m      e
3 rows selected

SQL> select m.id, m.m4, r.r4 from mesa m inner join rasa r on (m.id = r.id);
      1  m      e
1 row selected

! Problem #2:
!
! The following query using reverse scan returns 2 rows incorrectly on Alpha
! but 3 rows correctly on IA64:
!

SQL> select id, r4 from rasa where id = 1 and r4 <= 'm' order by id, r4;
Tables:
  0 = RASA
Index only retrieval of relation 0:RASA
Index name  Y4 [2:1]  Reverse Scan
Keys: (0.ID = 1) AND (0.R4 <= 'm')
  ID  R4
   1  k
   1  m
2 rows selected

SQL> select id, r4 from rasa where id = 1 and r4 <= 'm' order by id, r4;
Tables:
  0 = RASA
Index only retrieval of relation 0:RASA
Index name  Y4 [2:1]  Reverse Scan
Keys: (0.ID = 1) AND (0.R4 <= 'm')
  ID  R4
   1  e
   1  k
   1  m
3 rows selected
```

This problem is related to the construction and comparison of the descending key values while processing the index partitions.

The problem will be corrected in a future version of Oracle Rdb.

15.1.5 Remote Attach Stalls Before Detecting a Node is Unreachable

Bug 7681548

A remote attach can stall for a noticeable period, typically 75 seconds, before detecting a node is unreachable.

The following example shows the expected error message when attempting to access a database on a node that is not reachable. The problem is that when the value of the parameter `SQL_NETWORK_TRANSPORT_TYPE` in the file `RDB$CLIENT_DEFAULTS.DAT` is not specifically set to `DECNET` (in UPPER CASE), a stall of typically 75 seconds will happen before you get the expected error message.

```
SQL> attach 'file 1::disk1:[dbdir]db';
%SQL-F-ERRATTDEC, Error attaching to database 1::disk1:[dbdir]db
-RDB-F-IO_ERROR, input or output error
-SYSTEM-F-UNREACHABLE, remote node is not currently reachable
```

There are two possible ways to avoid the stall and get the error message after a user configurable period of time or instantly: decrease the value of the TCPIP parameter `TCP_KEEPINIT`, or explicitly specify `SQL_NETWORK_TRANSPORT_TYPE` as `DECNET` (in UPPER CASE).

- The default behavior when attempting to connect to an unreachable node via TCPIP is to stall 75 seconds before returning an error. The stall time is configurable in TCPIP via the parameter `TCP_KEEPINIT` which is expressed in units of 500 ms. The default value of `TCP_KEEPINIT` is 150 which corresponds to a 75 second stall.
- When connecting via DECnet, the error message is typically returned instantly so a significant stall will not be seen in this case. However, the value of the parameter `SQL_NETWORK_TRANSPORT_TYPE` is case sensitive so for DECnet to be selected as the transport, "DECNET" must be specified in UPPER CASE. Failing to do so will result in connecting via the DEFAULT method which is to first try connecting via DECnet and if that fails attempt to connect via TCPIP and hence a 75 second stall will take place unless `TPC_KEEPINIT` is set to a value lower than 150.

15.1.6 Application and Oracle Rdb Both Using SYS\$HIBER

In application processes that use Oracle Rdb and the `SYS$HIBER` system service (possibly via RTL routines such as `LIB$WAIT`), it is very important that the application ensures that the event being waited for has actually occurred. Oracle Rdb utilizes `$HIBER/$WAKE` sequences for interprocess communication and synchronization.

Because there is just a single process-wide "hibernate" state along with a single process-wide "wake pending" flag, Oracle Rdb must assume that it "shares" use of the hibernate/wake state with the user's application code. To this end, Oracle Rdb generally will re-wake the process via a pending wake request after using a hibernate sequence.

Oracle Rdb's use of the \$WAKE system service will 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, applications that use HIBER/WAKE facilities must use a code sequence that avoids continuing without a check for the operation (such as a delay or a timer firing) being complete.

The following pseudo-code shows one example of how a flag can be used to indicate that a timed-wait has completed correctly. The wait does not complete until the timer has actually fired and set TIMER_FLAG to TRUE. This code relies on ASTs being enabled.

```
ROUTINE TIMER_WAIT:
  BEGIN
    ! Clear the timer flag
    TIMER_FLAG = FALSE

    ! Schedule an AST for sometime in the future
    STAT = SYS$SETIMR (TIMADR = DELTATIME, ASTRTN = TIMER_AST)
    IF STAT <> SS$_NORMAL THEN LIB$SIGNAL (STAT)

    ! Hibernate. When the $HIBER completes, check to make
    ! sure that TIMER_FLAG is set indicating that the wait
    ! has finished.
    WHILE TIMER_FLAG = FALSE
    DO SYS$HIBER()
    END

ROUTINE TIMER_AST:
  BEGIN
    ! Set the flag indicating that the timer has expired
    TIMER_FLAG = TRUE

    ! Wake the main-line code
    STAT = SYS$WAKE ()
    IF STAT <> SS$_NORMAL THEN LIB$SIGNAL (STAT)
  END
```

Starting with OpenVMS V7.1, the LIB\$WAIT routine includes a FLAGS argument (with the LIB\$_NOWAKE flag set) to allow an alternate wait scheme (using the \$SYNCH system service) that can avoid potential problems with multiple code sequences using the \$HIBER system service. See the OpenVMS RTL Library (LIB\$) Manual for more information about the LIB\$WAIT routine.

In order to prevent application hangs, inner-mode users of SYS\$HIBER must take explicit steps to ensure that a pending wake is not errantly "consumed". The general way of accomplishing this is to issue a SYS\$WAKE to the process after the event is complete if a call to SYS\$HIBER was done. Rdb takes this step and therefore application programs must be prepared for cases where a wakeup might appear unexpectedly.

15.1.7 Unexpected RCS Termination

It has been observed in internal testing of Rdb Release 7.2.2 that if the Record Cache Server (the RCS) terminates in an uncontrolled fashion this may, under some conditions, cause corruption of the database and/or the After Image Journal file.

When the RCS terminates, the database is shut down and a message like the following is written to the monitor log:

Oracle® Rdb for OpenVMS

```
6-DEC-2007 15:04:17.02 - Received Record Cache Server image termination from
22ED5144:1
- database name "device:[directory]database.RDB:1" [device] (1200,487,0)
- abnormal Record Cache Server termination detected
- starting delete-process shutdown of database:
  - %RDMS-F-RCSABORTED, record cache server process terminated abnormally
- sending process deletion to process 22ED10F9
- sending process deletion to process 22ECED59
- sending process deletion to process 22EC0158
- sending process deletion to process 22EB9543 (AIJ Log server)
- database shutdown waiting for active users to terminate
```

A future attempt to roll forward the AIJ following a restore of a database backup might fail with a bugcheck dump if this problem has happened.

The only currently known situation where this problem has been observed is if the logical name RDM\$BIND_RCS_VALIDATE_SECS is defined to some value and the logical name RDM\$BIND_RCS_LOG_FILE at the same time is undefined or defined incorrectly.

To prevent this problem, Oracle recommends any customer using the Row Cache feature to either avoid defining the logical name RDM\$BIND_RCS_VALIDATE_SECS or if this logical name for any reason needs to be defined, to make sure RDM\$BIND_RCS_LOG_FILE is correctly defined (i.e. defined with the /SYSTEM and /EXECUTIVE qualifiers and is pointing to a valid file name in an existing directory on a cluster accessible device with sufficient free space).

This recommendation applies to all versions of Oracle Rdb.

15.1.8 Changes for Processing Existence Logical Names

Oracle Rdb Release 7.2.1.1 changed the handling of so called "existence" logical names used to tune the Rdb environment. These existence logical names could in past versions be defined to any value to enable their effect. The Rdb documentation in most cases described using the value 1 or YES as that value and this change is upward compatible with the documentation.

Rdb now treats these logical names (see the list below) as Boolean logicals and accepts a string starting with "Y", "y", "T", "t" or "1" to mean TRUE. All other values will be considered to be FALSE. This change allows process level definitions to override definitions in higher logical name tables which was not possible previously.

Oracle recommends that customers examine all procedures that define the following logical names to ensure that their values conform to these rules prior to upgrading to Oracle Rdb V7.2.1.1 or later to avoid unexpected changes in behavior.

- RDM\$AUTO_READY
- RDM\$DISABLE_HIDDEN_KEY
- RDM\$DISABLE_MAX_SOLUTION
- RDM\$DISABLE_REVERSE_SCAN
- RDM\$DISABLE_TRANSITIVITY
- RDM\$DISABLE_ZIGZAG_BOOLEAN
- RDM\$ENABLE_BITMAPPED_SCAN
- RDM\$ENABLE_INDEX_COLUMN_GROUP
- RDM\$MAX_STABILITY

- RDMS\$USE_OLD_COST_MODEL
- RDMS\$USE_OLD_COUNT_RELATION
- RDMS\$USE_OLD_SEGMENTED_STRING
- RDMS\$USE_OLD_UPDATE_RULES

15.1.9 SQL Module or Program Fails with %SQL-F-IGNCASE_BAD

Bug 2351248

A SQL Module or Pre-compiled SQL program built with Rdb 6.1 or earlier may fail when running under Rdb 7.3 if the program submits queries that involve certain kinds of character operations on parameters in the queries. For example, a LIKE operator in the WHERE clause of a SQL statement requires SQL to look for character- or string-matching wildcard characters. Another example is the use of IGNORE CASE which causes SQL to equivalence upper and lower case characters for the character set in use.

The following example shows a portion of a SQL module language program that queries a PERSONNEL database.

```

DECLARE MANL_NAME_LIST CURSOR FOR
  SELECT DISTINCT E.LAST_NAME, E.FIRST_NAME, J.JOB_CODE, J.DEPARTMENT_CODE, E.CITY
FROM   DB1_HANDLE.EMPLOYEES E, DB1_HANDLE.JOB_HISTORY J
WHERE  J.EMPLOYEE_ID = E.EMPLOYEE_ID
      AND E.STATUS_CODE = STATUS_CODE
      AND E.CITY LIKE CITYKEY IGNORE CASE
      ORDER BY E.EMPLOYEE_ID DESC, E.LAST_NAME DESC

PROCEDURE SQL_OPN_NAME_LIST
SQLCODE
CITYKEY      CHAR(20)
STATUS_CODE  CHAR(1);
OPEN MANL_NAME_LIST;

```

If the SQL Module containing the code above is compiled and linked into an executable using a pre-7.0 version of Rdb, it will run properly against that version. However if the same program is run in an Rdb 7.3 or later environment, a call to the SQL_OPN_NAME_LIST procedure will return a SQLCODE of -1. The RDB\$MESSAGE_VECTOR will contain a code associated with the following message:

```
%SQL-F-IGNCASE_BAD, IGNORE CASE not supported for character set
```

To workaroud this problem, re-link the program using a 7.3 or later version of SQL\$INT.EXE and/or SQL\$USER.OLB.

15.1.10 External Routine Images Linked with PTHREAD\$RTL

The OpenVMS Guide to the POSIX Threads Library describes that it is not supported to dynamically activate the core run-time library shareable image PTHREAD\$RTL. Oracle has found in testing that a shareable image supplied for use as an External Routine that is linked with PTHREAD\$RTL can be expected to cause a hang during dynamic image activation on OpenVMS I64 systems. This problem has not been observed on

OpenVMS Alpha systems.

To avoid this problem, in any case where the shareable image used for an Rdb External Routine is linked with PTHREAD\$RTL, the main program image must likewise be linked with PTHREAD\$RTL. This requirement applies to customer built application main programs as well as the main interactive SQL image.

The shareable image RDB\$NATCONN_FUNC73.EXE supplied with OCI Services for Oracle Rdb (part of SQL/Services) is one such shareable image that is linked with PTHREAD\$RTL. Customer built applications that utilize External Routines from the RDB\$NATCONN_FUNC73.EXE image must ensure that the main image is linked with PTHREAD\$RTL. The external routines that a user may call that use functions from RDB\$NATCONN_FUNC73.EXE include:

- TO_CHAR
- TO_NUMBER
- TO_DATE

You can use the OpenVMS command ANALYZE/IMAGE to determine whether an image depends upon PTHREAD\$RTL. For more information, see the OpenVMS documentation.

15.1.11 Using Databases from Releases Earlier than V7.0

You cannot convert or restore databases earlier than the Oracle Rdb V7.0 format directly to Oracle Rdb V7.3 format. The RMU Convert command for Oracle Rdb V7.3 supports conversions from Oracle Rdb V7.0, V7.1 and V7.2 format databases only. If you have an Oracle Rdb V3.0 through V6.1 format database, you must convert it to at least Oracle Rdb V7.0 format and then convert it to Oracle Rdb V7.3 format. For example, if you have a V4.2 format database, you must convert it first to at least Oracle Rdb V7.0 format, then convert it to Oracle Rdb V7.3 format.

If you attempt to convert or restore a database that is prior to Oracle Rdb V7.0 format directly to Oracle Rdb V7.3 format, Oracle RMU generates an error.

15.1.12 ILINK-E-INVOCRINI Error on I64

When linking an application with multiple modules, the following error message may be returned:

```
%ILINK-E-INVOCRINI, incompatible multiple initializations for overlaid section
  section: VMSRDB
  module: M1
  file: DKA0:[BLD]M1.OBJ;1
  module: M2
  file: DKA0:[BLD]SYS.OLB;1
```

On I64 systems, it is not allowed to have a program section that attempts to be initialized a subsequent time where the non-zero portions of the initializations do not match. This is a difference from OpenVMS Alpha and VAX systems where the linker permitted such initializations.

If the modules specified are SQL module language or precompiler produced, the application build procedures usually need to be modified. Typically, the solution is to initialize the database handles in only one of the modules. The SQLMOD command line qualifiers /NOINITIALIZE_HANDLES and /INITIALIZE_HANDLES are used to specify whether or not alias definitions are coerced into alias references.

15.1.13 New Attributes Saved by RMU/UNLOAD 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-NOTUNLFILE, 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.

15.1.14 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.

15.1.15 Oracle Rdb and OpenVMS ODS–5 Volumes

OpenVMS Version 7.2 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.

15.1.16 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, a user has two tables T1 and T2, both with one column, and wishes to ensure that all values in table T1 exist in T2. Both tables have an index on the referenced field. The user could use a PRIMARY KEY constraint on T2 and a FOREIGN KEY constraint on T1.

```
SQL> alter table t2 alter column f2 primary key not deferrable;
SQL> alter table t1 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 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 alter column f1
cont>  check (f1 in (select * from t2 where f2=f1)) not deferrable;

or:

SQL> alter table t1 alter column f1
cont>  check (f1=(select * from t2 where f2=f1)) 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 after insert on t1
cont> when (not exists (select * from t2 where f2=f1))
cont> (error) for each row;
SQL> create trigger t1_update 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 before delete on t2
cont> when (exists (select * from t1 where f1=f2))
cont> (error) for each row;
SQL> create trigger t2_modify 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.

15.1.17 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.

15.1.18 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.

15.1.19 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.

15.1.20 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

Oracle® Rdb for OpenVMS

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 SORT32 run-time library.

At the beginning of a sort operation, the SORT code allocates 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 process' quotas or parameters when allocating memory. The effects of WSQUOTA and WSEXTENT are indirect. At the beginning of each sort operation, the SORT code attempts to adjust the process working set to the maximum possible size using the \$ADJWSL system service specifying a requested working set limit of %X7FFFFFFF pages (the maximum possible). SORT 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 36. The work files can be individually controlled by the SORTWORKn logical names (where n ranges from "0" through "Z"). 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 36 logical names, "SORTWORK0" through "SORTWORKZ".

Normally, SORT places work files in the SYS\$SCRATCH directory. By default, SYS\$SCRATCH is the same device and directory as the SYS\$LOGIN location. Spreading the I/O load over multiple disks and/or controllers improves efficiency as well as performance by taking advantage of more system resources and helps prevent disk I/O bottlenecks. Specifying that 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

SYSSCRATCH disk device (for example, while Oracle Rdb builds indexes for a very large table). Using the "SORTWORK0" through "SORTWORKZ" 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 36 sort work files, it is possible to exceed the capacity of the first SORT file device and the sort operation fails never having accessed the remaining 35 sort work files.

At this time, more than 10 sort work files will only be used by the Oracle Rdb sort interface as used by the CREATE INDEX, ALTER INDEX and the clauses UNION DISTINCT, ORDER BY, GROUP BY and SELECT DISTINCT. The RMU and SQL IMPORT interfaces use the OpenVMS SORT interface which does not currently support more than 10 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.

15.1.21 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 15–1 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 very large working set limits can sometimes experience significant page faulting and CPU consumption while initializing sort memory. This logical name can restrict the sort work memory to a percentage of the 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 2147483647 and the minimum value is 32768.

15.1.22 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' 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 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.

15.2 SQL Known Problems and Restrictions

This section describes known problems and restrictions for the SQL interface.

15.2.1 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.

15.2.2 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:

```
$ 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.

15.2.3 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.

Oracle® Rdb for OpenVMS

This is not a bug; it is a restriction resulting from the way OpenVMS image activation works.

15.3 Oracle RMU Known Problems and Restrictions

This section describes known problems and restrictions for the RMU interface.

15.3.1 RMU/CONVERT Fails When Maximum Relation ID is Exceeded

If, when relation IDs are assigned to new system tables during an RMU/CONVERT to a V7.2 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 the prior database version. 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-CVTROLSUC error message when the database has been rolled back to V7.0 since it cannot be converted to V7.2:

```
$rmu/convert mf_personnel
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.2
Are you satisfied with your backup of
  DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 and your backup of
  any associated .aij files [N]? Y
%RMU-I-LOGCONVRT, database root converted to current structure level
%RMU-F-RELMAXIDBAD, ROLLING BACK CONVERSION - Relation ID exceeds maximum
  8192 for system table RDB$RELATIONS
%RMU-F-CVTROLSUC, 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 V7.2
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 V7.2
%RMU-I-CVTCOMSUC, CONVERT committed for
  DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 to version V7.2
```

15.3.2 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.

15.3.3 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.

15.3.4 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.

15.3.5 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.

15.3.6 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.

15.4 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.

15.4.1 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 maximum of about 600 disk blocks. The actual increase depends mostly on the maximum number of users specified for the database.

15.4.2 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.

15.4.3 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.

15.4.4 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 illustrates the behavior:

```
SQL> create table T1
cont>      (id integer
cont>        ,last_name char(12)
cont>        ,first_name char(12)
cont>        );
SQL> create storage map M for T1
```

```

cont>      partitioning not updatable
cont>      store using (id)
cont>          in EMPIDS_LOW with limit of (200)
cont>          in EMPIDS_MID with limit of (400)
cont>          otherwise in EMPIDS_OVER;
SQL> insert into T1 values (150,'Boney','MaryJean');
1 row inserted
SQL> insert into T1 values (350,'Morley','Steven');
1 row inserted
SQL> insert into T1 values (300,'Martinez','Nancy');
1 row inserted
SQL> insert into T1 values (450,'Gentile','Russ');
1 row inserted
SQL>
SQL> set flags 'EXECUTION(100),STRATEGY,DETAIL(2),INDEX_PARTITIONS';
SQL>
SQL> select * from T1 where ID > 400;
~S#0001
Tables:
  0 = T1
Conjunct: 0.ID > 400
Get      Retrieval sequentially of relation 0:T1          (partitioned scan#1)
~E#0001.1: Strict Partitioning using 2 areas
      partition 2 (larea=60) "EMPIDS_MID"
      partition 3 (larea=61) "EMPIDS_OVER" otherwise
              ID  LAST_NAME  FIRST_NAME
              450  Gentile   Russ
1 row selected
SQL>

```

In this example, partition 2 does not need to be scanned but is still accessed due to the structure of the generated key values. This does not affect the correctness of the result. Users can avoid the extra scan by using values other than the boundary values.

15.4.5 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 the following errors:

```
%RDMS-F-NOEUACCESS, unable to acquire exclusive access to database
```

or

```

%RDB-F-SYS_REQUEST, error from system services request
-RDMS-F-FILACCERR, error opening database root DKA0:[RDB]TEST.RDB;1
-SYSTEM-W-ACCONFLICT, file access conflict

```

You can make this change only when no users are attached to the database and, if the database is set to OPEN IS MANUAL, the database is closed. Several internal Oracle Rdb data structures are based on the minimum page size and these structures cannot be resized if users are attached to the database.

Furthermore, because this particular change is not recorded in the AIJ, 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.

15.4.6 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.

15.4.7 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.

15.5 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.

15.5.1 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.

15.5.2 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 15–2 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

15.5.3 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>
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.

15.5.4 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.

15.5.5 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](#)