

Oracle® Rdb for OpenVMS

New Features Manual

Release 7.3

July 2023

ORACLE®

Oracle Rdb New Features, Release 7.3.4 for OpenVMS

Copyright © 1984, 2023 Oracle and/or its affiliates. All rights reserved.

Oracle Corporation - Worldwide Headquarters, 2300 Oracle Way, Austin, TX 78741, United States

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, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs) and Oracle computer documentation or other Oracle data delivered to or accessed by U.S. Government end users are "commercial computer software" or "commercial computer software documentation" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, reproduction, duplication, release, display, disclosure, modification, preparation of derivative works, and/or adaptation of i) Oracle programs (including any operating system, integrated software, any programs embedded, installed or activated on delivered hardware, and modifications of such programs), ii) Oracle computer documentation and/or iii) other Oracle data, is subject to the rights and limitations specified in the license contained in the applicable contract. The terms governing the U.S. Government's use of Oracle cloud services are defined by the applicable contract for such services. No other rights are granted to the U.S. Government.

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.

Intel and Intel Inside are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Epyc, and the AMD logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

Contents

Preface	v
1 Enhancements and Changes Provided in Oracle Rdb Release 7.3.4	
1.1 Enhancements and Changes Provided in Oracle Rdb Release 7.3.4	1-1
1.1.1 Enhanced LIKE Table Support in CREATE TABLE Statement	1-1
1.1.2 Some Aggregate Functions Now Inherit Source Column EDIT STRING	1-1
1.1.3 New Summary_Only Qualifier to RMU Dump Audit Command	1-2
1.1.4 New Option=GENERATED Added to RMU Extract Command	1-3
1.1.5 Changed Behavior for the Noedit_Filename Qualifier in RMU Backup After_Journal Command	1-4
2 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.2	
2.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.2	2-1
2.1.1 New PCSI Support for Rdb Kit Installation and Deinstallation	2-1
2.1.2 Updated Support for RDMSS\$BIND_CODE_OPTIMIZATION	2-2
2.1.3 DUPLICATES ARE NOT ALLOWED Clause Added to ALTER INDEX Statement	2-3
2.1.4 New RMU/SET_DATABASE Qualifier /ACCESS=[UN]RESTRICTED	2-3
2.1.5 New RMU/VERIFY Qualifier /VALIDATE for FLOAT Data Type Error Detection	2-4
2.1.6 New /ABORT Qualifier for the RMU/SET DATABASE Command ...	2-6
2.1.7 Smaller After Image Backup Files	2-8
3 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.1	
3.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.1	3-1
3.1.1 New and Modified Replication Network Error Handling Log Messages	3-1
3.1.2 RMU Show Statistics Feature To Select Screens To Include In STATISTICS.RPT	3-2
3.1.3 RMU BACKUP and RESTORE /OUTPUT Qualifier to Write Output to a File	3-4
3.1.4 AIJ Backup Data Compression Information is Now in the ABS Process Logs	3-5
3.1.5 RMU MOVE_AREA and COPY_DATABASE /OUTPUT Qualifier to Write Output to a File	3-6
3.1.6 Support for /OUTPUT=file-spec Qualifier Added to RMU Backup Plan File	3-7
3.1.7 New CREATE OR REPLACE Support for PROFILE	3-9
3.1.8 New CREATE OR REPLACE Support for VIEW	3-10

3.1.9	New CREATE OR REPLACE Support for SEQUENCE	3-12
3.1.10	New Options for SET LOGFILE Statement	3-13
3.1.11	New UNDECLARE CURSOR Statement	3-14
3.1.12	Enhanced LIKE Table Support in CREATE TABLE Statement	3-15
3.1.13	New TO_DSINTERVAL and TO_YMINTERVAL Functions	3-18
3.1.14	New CREATE DEFAULT AUDIT Statement	3-20
3.1.15	New ALTER DEFAULT AUDIT Statement	3-23
3.1.16	New DROP DEFAULT AUDIT Statement	3-25
3.1.17	New SESSION and GLOBAL Attributes for Sequences	3-26
3.1.18	New LogMiner Feature to Close and Immediately Reopen Table Output Files	3-27

4 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.0

4.1	Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.0	4-1
4.1.1	Intel Itanium Processor 9700 “Kittson” Certified	4-1
4.1.2	Relaxed Type Checking for DEFAULT Clause	4-1
4.1.3	New Statistics Screen Shows Top Processes Accessing a Table Logical Area	4-2
4.1.4	Relaxed Naming Rules for RMU Extract Option=MATCH Option	4-4
4.1.5	RMU/RESTORE Now Always Displays the %RMU-I-AIJREFCFL Message	4-5
4.1.6	New SQL Built-in Functions	4-6
4.1.6.1	New String Functions	4-6
4.1.6.2	New Aggregate Functions	4-8
4.1.7	-RMU-F-DBROOTFILE, -RMU-F-DBDATAFILE messages output with %RMU-F-BADAIJFILE	4-12
4.1.8	RMU Extract Now Outputs ALTER DATABASE For Storage Area Access Mode	4-13
4.1.9	RMU/RECOVER RMU-F-BACKUPNOAIJ, RMU-F-TSNNOSYNC, RMU-F-CANTSYNCTSNS Error Messages	4-14
4.1.10	Delimited_Text Keywords Can Now Be Negated For RMU Load And Unload	4-17
4.1.11	RMU Load Now Supports User Defined Conversion Routines	4-17
4.1.12	New CARDINALITY Option for SHOW TABLE Command	4-19
4.1.13	New CONSTRAINT Naming for Domain Constraints	4-19
4.1.14	New AS Result-type Clause for CREATE SEQUENCE Statement	4-20
4.1.15	New GENERATED Column Support	4-21
4.1.16	Enhancements to INCLUDE Statement	4-22
4.1.17	New Support for DEFAULT Index NODE SIZE Calculation	4-23
4.1.18	New LANGUAGE Support From RMU Extract Command	4-24
4.1.19	Enhancements for CREATE and ALTER MODULE Statements	4-26
4.1.20	New RMU Dump Symbols Command	4-28
4.1.21	New Options to SET SQLDA Statement	4-30
4.1.22	More New Options to SET SQLDA Statement	4-30

5 Enhancements And Changes Provided in Oracle Rdb Release 7.3.2.1

5.1	Enhancements And Changes Provided in Oracle Rdb Release 7.3.2.1	5-1
5.1.1	Oracle Rdb 7.3.2.1 Certified on OpenVMS 8.4-2 from VMS Software Inc. and Integrity i4 systems from HPE	5-1
5.1.2	RMU/SHOW AFTER_JOURNAL [NO]CHECKPOINT Qualifier	5-1
5.1.3	Engine Error Logging	5-4
5.1.4	New MEDIAN Aggregate Function Added to SQL	5-5

5.1.5	New RMU/BACKUP/AFTER_JOURNAL [NO]SPACE_CHECK Qualifier	5-5
5.1.6	New Options to SET SQLDA Statement	5-7
5.1.7	New RMU Set Statistics Command	5-8
5.1.8	Multi-Aggregate Index Optimization	5-13
5.1.9	Use Old DPB Format for Rdb_Change_Database	5-15
5.1.10	LogMiner State Now in AIJ Options File, New RDM\$LOGMINER_STATE Symbol	5-15
5.1.11	New /[NO]MBX_ASYNCH Qualifier for RMU/UNLOAD/AFTER	5-21
5.1.12	New /PAGE_NUMBER Qualifier for RMU/DUMP and RMU/DUMP/BACKUP	5-21
5.1.13	New Information Table RDB\$SESSION_PRIVILEGES Now Available	5-22

6 Enhancements And Changes Provided in Oracle Rdb Release 7.3.2.0

6.1	Enhancements And Changes Provided in Oracle Rdb Release 7.3.2.0	6-1
6.1.1	New COMPRESSION OCTETS Clause for CREATE INDEX Statement	6-1
6.1.2	Enhancements for TRUNCATE TABLE Statement	6-2
6.1.3	RMU Extract Now Supports RECOMPILE Item	6-4
6.1.4	SQL Now Supports the MISSING VALUE Clause as Part of CREATE and ALTER DOMAIN Statement	6-4
6.1.5	Comma Statement Separator Now Deprecated	6-5
6.1.6	New Logical Name RDMSS\$BIND_DEADLOCK_WAIT to Control Sub-second Deadlock Wait	6-6
6.1.7	Query Optimization Improvements for IN Clause	6-7
6.1.8	New SHOW AUDIT Command Added to Interactive SQL	6-8
6.1.9	RMU/RECLAIM Can Now Skip to the Next SPAM Interval and/or Storage Area to Avoid Lock Contention	6-9
6.1.10	RMU Open Statistics Supports PROCESS_GLOBAL Qualifier	6-11
6.1.11	RMU/SHOW LOGICAL_NAME Now Supports /DESCRIPTION Qualifier	6-12
6.1.12	Using Per-Process Monitoring for RMU Show Statistics	6-13
6.1.12.1	Per-Process Monitoring Operational Modes	6-13
6.1.12.2	Per-Process Monitoring Facility Activation	6-14
6.1.12.3	Per-Process Monitoring Facility Process Activation	6-15
6.1.12.4	Per-Process Monitoring Run-Time Options	6-17
6.1.12.5	Detached Process Monitoring	6-18
6.1.12.6	Per-Process Monitoring Overview Information	6-18
6.1.13	RMU Error Messages Which Suggest Altering Backup File Attributes	6-27
6.1.14	Query Optimization Improvements for DATE ANSI Queries	6-29
6.1.15	New RMU Dump Metadata_File Command	6-29
6.1.16	New REPLACE_ROWS Qualifier Added to RMU Load Command	6-31
6.1.17	RMU/SET SHARED_MEMORY/SECTION_NAME	6-31
6.1.18	RESTART Clause of ALTER SEQUENCE No Longer Needs Value	6-33
6.1.19	New Options to SET SQLDA Statement	6-33
6.1.20	Support for External Authentication (LDAP)	6-34
6.1.21	New RMU Extract Options to Control Output for DATABASE and ALTER_DATABASE Items	6-35
6.1.22	RMU/SET AFTER_JOURNAL/SWITCH_JOURNAL Now Can Create an Emergency AIJ	6-36
6.1.23	New Support for Second OpenVMS Account Password	6-39

6.1.24	New "Index Counts" Optimization for SORTED Indices	6-40
6.1.25	Support for Proxy Access to Remote Databases Using TCP/IP Transport	6-40
6.1.26	Support for INTEGER Result Type for COUNT Function	6-41
7	Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.3	
7.1	Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.3	7-1
7.1.1	Oracle Rdb 7.3.1.3 Certified on OpenVMS 8.4-1H1 from VMS Software Inc. and Integrity i4 systems from HPE	7-1
8	Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.2	
8.1	Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.2	8-1
8.1.1	New FULBCKREQ Message Output When a Full Backup is Required	8-1
8.1.2	New TRACE Option for EXPORT DATABASE Statement	8-2
8.1.3	New /NOAFTER_JOURNAL Qualifier to Disable AIJ File Creation by RMU/RECOVER	8-3
8.1.4	Enhance Dumper of Merge Range List	8-4
8.1.5	RMU Extract Now Extracts SYS_GET_DIAGNOSTIC Function	8-5
8.1.6	Alter Index Now Supports REVERSE and NOREVERSE Clauses	8-5
8.1.7	SQL Precompiler Now Generates C++ Compatible Intermediate C Source	8-6
8.1.8	New RMU/RECOVER Feature to Alter Storage Areas, After_Journal and Record_Cache Directories	8-7
8.1.9	RMU Unload Record_Definition File Can Include Offset and Length Comment	8-14
8.1.10	New RMU/DUMP/BACKUP Enhanced Error Handling Features	8-15
8.1.11	New REVERSE Attribute for CREATE SEQUENCE Statement and IDENTITY Clause	8-16
9	Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.1	
9.1	Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.1	9-1
9.1.1	New LIMIT_TO Qualifier Added to RMU Load Command	9-1
9.1.2	New BEFORE and SINCE Qualifiers Added to RMU Load Audit	9-2
9.1.3	New RMU/SHOW/STATISTICS Output File Periodic Buffer Flushes	9-3
9.1.4	New Error and Log Messages Added for Segmented String Verification	9-4
10	Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.0	
10.1	Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.0	10-1
10.1.1	Changes to Default and Limits Behavior in Oracle Rdb	10-1
10.1.2	New /ERROR_LIMIT Qualifier Added as the Default to RMU/VERIFY	10-3
10.1.3	RMU /VERIFY Root Displays the Corrupt Page Table Entries	10-5
10.1.4	DECLARE LOCAL TEMPORARY TABLE Supports COMMENT IS Clause	10-6
10.1.5	Temporary Tables Now Support LARGE MEMORY Option	10-6
10.1.6	COUNT Now Returns BIGINT Result	10-7

10.1.7	Aggregate Functions Now Use BIGINT Counters	10-7
10.1.8	/[NO]KEY_VALUES Qualifier Added to RMU/VERIFY/INDEX	10-7
10.1.9	The /LOCK_TIMEOUT Qualifier Now Allows the Database Default	10-9
10.1.10	Compression of AIJ Backup Files for Automatic AIJ Backups	10-10
10.1.11	Global Statistics Sections for Better Performance	10-10
10.1.12	RMU/SET AUDIT Supports Wildcard Table and Column Names	10-10
10.1.13	RMU/BACKUP Database Root Verification Performance Enhancement	10-12
10.1.14	RMU /UNLOAD /AFTER_JOURNAL New Qualifier /DELETES_FIRST	10-14
10.1.15	Add Option to Pass Values to /CONFIRM During RESTORE Operation	10-14
10.1.16	Table Names Can Now Be Specified For Index Verification	10-15
10.1.17	New RMU/VERIFY Feature to Detect Orphan Hash Index Buckets	10-16
10.1.18	New COMPILE Clause for ALTER TRIGGER Statement	10-17
10.1.19	New COMPILE ALL TRIGGERS Clause for ALTER TABLE Statement	10-18
10.1.20	New RETRY Clause for ACCEPT Statement	10-20
10.1.21	New Character Sets ISOLATIN2 and WIN_LATIN2 Supported	10-20
10.1.22	Changes and Enhancements to Trigger Support	10-21
10.1.23	New RMU BACKUP RBF File BRH\$K_ROOT1, BRH\$K_ROOT2, BRH\$K_ROOT3 Records /kroot_records	10-21
10.1.24	New Functions NUMTODSINTERVAL, NUMTOYMINTERVAL Supported	10-23
10.1.25	RMU Dump Audit Command	10-24
10.1.26	New BIN_TO_NUM Numeric Function	10-29
10.1.27	RMU /PROGRESS_REPORT and Control-T for RMU Backup and Restore	10-29
10.1.28	/[NO]SNAPSHOTS, /[NO]DATA_FILE Added to RMU/MOVE_AREA	10-30
10.1.29	Enhancements for Compression Support in SQL EXPORT DATABASE Command	10-32
10.1.30	/[NO]PARTITIONS Qualifier Added to RMU/ANALYZE/INDEXES	10-33
10.1.31	/[NO]PARTITIONS Qualifier Added to RMU/ANALYZE Storage Statistics	10-37
10.1.32	/[NO]PARTITIONS Qualifier Added to RMU/ANALYZE/PLACEMENT	10-42
10.1.33	New RMU/[NO]ASSIST Qualifier for Commands Using Tape Drives	10-47
10.1.34	New RMU/ALTER Feature to Modify the Area Header Root File Specification	10-48
10.1.35	Create Index Supports the REVERSE Keyword to Create Reverse Key Indices	10-50
10.1.36	Support for New Syntax for Sequence Generator Statements	10-50
10.1.37	RMU/SET AUDIT Supports MODULE, ROUTINE, SEQUENCE and VIEW	10-52
10.1.38	RMU/SHOW AUDIT Supports MODULE, ROUTINE, SEQUENCE and VIEW	10-53
10.1.39	SQL Now Supports SQL Standard Syntax for SET CONSTRAINTS ALL Statement	10-54
10.1.40	Support ANSI and ISO SQL Standard Length Units	10-54

10.1.41	New SET FLAGS Clause Supported by CREATE and ALTER PROFILE	10-55
10.1.42	New Support for SAVEPOINT Syntax and Semantics	10-56
10.1.42.1	SAVEPOINT Statement	10-57
10.1.42.2	RELEASE SAVEPOINT Statement	10-58
10.1.42.3	ROLLBACK TO SAVEPOINT Statement	10-59
10.1.43	New OPTIMIZE OUTLINE Clause Allows Outline Specification	10-61
10.1.44	RMU/DUMP/HEADER=ROW_CACHE Now Displays Whether Row Cache is Enabled	10-62
10.1.45	RMU/LOAD Now Supports CSV Formatted Files	10-67
10.1.46	RMU/UNLOAD Now Supports CSV Formatted Files	10-67
10.1.47	RMU/UNLOAD Supports BITMAPPED_SCAN Optimize Option	10-69
10.1.48	New EDIT STRING Clause for CREATE FUNCTION and CREATE MODULE Functions	10-69
10.1.49	Changes to RMU/VERIFY/CONSTRAINTS and ALTER TABLE Statement	10-71
10.1.50	New SQRT Numeric Function	10-71
10.1.51	New MOD Numeric Function	10-72
10.1.52	New Data Types BINARY and BINARY VARYING	10-74
10.1.53	PERSONA SUPPORT is Enabled For All New Databases	10-75
10.1.54	New Dialects Support in SQL	10-76
10.1.55	New WITH Clause Provides Subquery Factoring	10-77
10.1.56	DECLARE LOCAL TEMPORARY VIEW Statement	10-80
10.1.57	Enhancements for Buffered Read Support in SQL EXPORT DATABASE Command	10-81
10.1.58	New BITMAPPED SCAN Clauses Added to OPTIMIZE Clause	10-82
10.1.59	New Support for Allocations Specified Using Quantified Numeric Literal	10-84
10.1.60	New SQL Functions Added	10-85
10.1.61	Optimized NOT NULL Constraint Execution	10-85
10.1.62	New RMU/LOAD Option CHARACTER_ENCODING_XML	10-86
10.1.63	New MEMORY ALLOCATION Clause for the GLOBAL BUFFERS Definition	10-87
10.1.64	New REPLACE Statement	10-87
10.1.65	Query Optimization Improvements for IN Clause	10-89

11 Optimizer Enhancements Appendix

11.1	Optimizer Enhancements	11-1
11.1.1	Changes and Improvements to the Rdb Optimizer and Query Compiler	11-1

Examples

4-1	Using the FIRST_VALUE Function	4-8
4-2	Using the LAST_VALUE Function	4-9
4-3	Using the LISTAGG Function	4-11
4-4	Using the GROUP_CONCAT Function	4-11
4-5	Using RMU Dump Symbols	4-29
5-1	RMU Set Statistics Export	5-12
5-2	RMU Set Statistics Checkpoint	5-13
6-1	Example showing use of PROCESS_GLOBAL option	6-12

6-2	Using the STORAGE_AREAS and JOURNALS options to control output	6-35
6-3	Using MATCH option to extract just one row cache definition	6-36
10-1	Using CSV format for Microsoft EXCEL export	10-68
10-2	Using options to change delimiters in a CSV formatted file	10-68
10-3	Example 1: Invalid request for square root of a negative value	10-72
10-4	Example 2: Correct query showing square root results	10-72
10-5	Example 1: Using the MOD function	10-73
10-6	Example 1: Using the old syntax vs the new syntax for the WITH clause	10-79
10-7	Example 2: Using Complex Query with INSERT ... SELECT Statement	10-79
10-8	Example 3: Using subquery factoring within a UNION operator	10-80
10-9	Example 1: Simplifying a query using a declared local view	10-82
10-10	Example 2: Operations on an updatable local view	10-83

Tables

10-1	RDBNSASK types	10-26
------	----------------------	-------

Purpose of This Manual

This manual contains the New Features Chapters for Oracle Rdb Release 7.3.4 and prior Rdb 7.3 releases.

Deprecated and Desupported Features for Oracle Rdb

Each release of Oracle Rdb introduces behavior changes for your database in addition to new features. Changes in behavior include deprecated and desupported debug flags, parameters, options, syntax, and the deprecation and desupport of features and components.

Each chapter in this manual describes behavior changes where features have been deprecated or desupported in that release. By deprecate, we mean that the feature is no longer being enhanced but is still supported for the full life of the Oracle Rdb release. By desupported, we mean that Oracle will no longer fix bugs related to that feature and may remove the code altogether (see the *Obsolete Features* section in each chapter). Where indicated, a deprecated feature may be desupported in a future major release.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Document Structure

This manual consists of the following chapters:

Chapter 1	Describes enhancements introduced in Oracle Rdb Release 7.3.4
Chapter 2	Describes enhancements introduced in Oracle Rdb Release 7.3.3.2
Chapter 3	Describes enhancements introduced in Oracle Rdb Release 7.3.3.1
Chapter 4	Describes enhancements introduced in Oracle Rdb Release 7.3.3.0
Chapter 5	Describes enhancements introduced in Oracle Rdb Release 7.3.2.1
Chapter 6	Describes enhancements introduced in Oracle Rdb Release 7.3.2.0
Chapter 7	Describes enhancements introduced in Oracle Rdb Release 7.3.1.3
Chapter 8	Describes enhancements introduced in Oracle Rdb Release 7.3.1.2
Chapter 9	Describes enhancements introduced in Oracle Rdb Release 7.3.1.1
Chapter 10	Describes enhancements introduced in Oracle Rdb Release 7.3.1.0
Chapter 11	Describes enhancements in the Optimizer

Enhancements and Changes Provided in Oracle Rdb Release 7.3.4

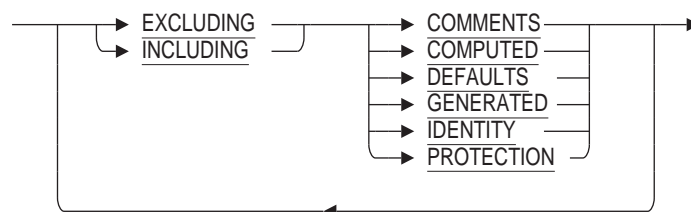
1.1 Enhancements and Changes Provided in Oracle Rdb Release 7.3.4

1.1.1 Enhanced LIKE Table Support in CREATE TABLE Statement

This release of Oracle Rdb adds new EXCLUDING and INCLUDING clauses to the LIKE clause within the CREATE TABLE statement.

Syntax

like-attributes =



By default, Rdb includes the column protections (access control lists) and comments for any copied column. These new clauses allow the database administrator to suppress the copying of that metadata.

1.1.2 Some Aggregate Functions Now Inherit Source Column EDIT STRING

Oracle Rdb V7.3.3 and later versions support EDIT STRING inheritance for these functions when using Interactive SQL.

- MAX, MEDIAN, MIN, FIRST_VALUE, LAST_VALUE

When the input type matches the output type, then the EDIT STRING from the source column is inherited to improve the readability of the aggregate.

- CAST

When the datatype of the CAST function references a domain then if an EDIT STRING defined it is inherited from the domain.

The following example shows the EDIT STRING being used.

```

SQL> create domain DOM_TST integer(2) edit string '$(9)9.99';
SQL>
SQL> create table TST
cont>     (a integer(2) edit string '$(9)9.99'
cont>     ,c char(10)
cont>     );
SQL>
SQL> insert into TST
cont>     values (100, 100, 'A');
1 row inserted
SQL> insert into TST
cont>     values (233, 233, 'B');
1 row inserted
SQL>
SQL> --> column with explicit edit string
SQL> select min (a), max (a), cast (a as DOM_TST)
cont> from TST
cont> group by a
cont> ;
cont> ;

           $100.00           $100.00           $100.00
           $233.00           $233.00           $233.00
2 rows selected
SQL>
SQL> select first_value (a) within group (order by b desc),
cont>         last_value (a) within group (order by b desc),
cont>         median (a)
cont> from TST
cont> ;

           $233.00           $100.00           $166.50
1 row selected
SQL>

```

Use the SET DISPLAY NO EDIT STRING statement to disable this behavior.

1.1.3 New Summary_Only Qualifier to RMU Dump Audit Command

This release of Oracle Rdb adds a new Summary_Only qualifier to RMU Dump Audit. This allows the database administrator to see a list of databases that have entries recorded in the named AUDIT\$JOURNAL.

Neither the Format nor the Type qualifiers are permitted when Summary_Only is used. The database parameter is ignored.

The following example generates a file containing the database names used by that version of the SECURITY.AUDIT\$JOURNAL.

```

$ define/nolog RMU_AJ SYS$COMMON: [SYSMGR]SECURITY.AUDIT$JOURNAL;8398
$ rmu/dump/audit -
  "" -
  RMU_AJ -
  /since=TODAY -
  /log -
  /summary_only -
  /output=audit_dump.txt
$

```

1.1.4 New Option=GENERATED Added to RMU Extract Command

This release of Oracle Rdb includes a new GENERATED option for RMU Extract. In prior releases RMU Extract Item=UNLOAD and Item=LOAD would generate load commands that assumed all the columns were updatable.

The option FULL can be used to generate syntax that loaded every field by name and included virtual columns (AUTOMATIC AS, GENERATED, IDENTITY and COMPUTED BY) as commented out names. Therefore, editing was required to uncomment GENERATED column names so they could be reloaded. In addition the /Virtual=AUTOMATIC qualifier needed to be added to the RMU Load and RMU Unload commands.

Using Option=GENERATED will now instruct RMU Extract to generate more appropriate DCL commands for unloading and re-loading data in tables that contain GENERATED columns.

The following example shows a portion of a generated DCL procedure when only Option=FULL is used.

```
$ RMU/EXTRACT/ITEM=UNLOAD/OPTION=FULL SAMPLE_DB
.
.
.
$ CREATE SAMPLE0.COLUMNNS
! Columns list for table SAMPLE0
! in ...
! Created by RMU Extract for Oracle Rdb ... on 29-JAN-2021 13:20:28.40
! Virtual: IDENT_COL
DETAILS
! Virtual: LAST_UPDATE
$ RMU/UNLOAD -
  USER1:[TESTING.DATABASES]MF_PERSONNEL_SQL.RDB -
  /FIELDS="@SAMPLE0.COLUMNNS" -
  SAMPLE0 -
  SAMPLE0.UNL
$
```

The following example shows a portion of a generated DCL procedure when Option=GENERATED is used.

```
$ RMU/EXTRACT/ITEM=UNLOAD/OPTION=GENERATED SAMPLE_DB
.
.
.
$ CREATE SAMPLE0.COLUMNNS
! Columns list for table SAMPLE0
! in ...
! Created by RMU Extract for Oracle Rdb ... on 29-JAN-2021 13:23:27.76
IDENT COL
DETAILS
LAST_UPDATE
$ RMU/UNLOAD -
  USER1:[TESTING.DATABASES]MF_PERSONNEL_SQL.RDB -
  /FIELDS="@SAMPLE0.COLUMNNS" -
  /VIRTUAL=AUTOMATIC -
  SAMPLE0 -
  SAMPLE0.UNL
$
```

1.1.5 Changed Behavior for the Noedit_Filename Qualifier in RMU Backup After_Journal Command

Bug 31711278

In prior releases of Oracle Rdb the Noedit_Filename qualifier on the RMU Backup After_Journal was ignored. With this release it takes on a new meaning as described below:

– /EDIT_FILENAME

As with previous versions, this qualifier defines the editing to be performed for the output backup file name. This editing is performed on the provided backup filename, or if "" is specified instead of a backup-file-spec then the default backup filename defined in the database.

This qualifier replaces any EDIT_FILENAME defined for the database.

– /NOEDIT_FILENAME

This qualifier negates any prior usage on the command of the /EDIT_FILENAME qualifier and also instructs RMU to ignore the EDIT_FILENAME defined by the SQL ALTER DATABASE statement, or RMU Set After_Journal command. This is a change of behavior from prior versions and supports the enhancements made to the RMU Set After_Journal command which allows the defaults to be defined for the MANUAL backup processing.

No editing is performed on the provided backup filename, or if "" is specified instead of a backup-file-spec then the default backup filename defined in the database is used without changes.

– Neither /EDIT_FILENAME nor /NOEDIT_FILENAME was used.

In this case RMU Backup After_Journal will use the default if defined in the database by SQL ALTER DATABASE statement, or RMU Set After_Journal command.

Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.2

2.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.2

2.1.1 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
-----
```

```

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 SYSSMANAGER: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

2.1.2 Updated Support for RDBMS\$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.

2.1.3 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.

2.1.4 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 `RMUSALTER` 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
```

2.1.5 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 -
        /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.

2.1.6 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
%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
$

```


2.1.7 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.

Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.1

3.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.1

3.1.1 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
```

3.1.2 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  
-----
```

3.1.3 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".

The default if this qualifier is not specified is to send output from the BACKUP or RESTORE command to SYS\$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

```

3.1.4 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
$
```

3.1.5 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 MOVTEXT_15, MOVTEXT_06, MOVTEXT_07, DOFULLBCK, -
  "operation completed"
%RMU-I-MOVTEXT_15, Area files have been moved
%RMU-I-MOVTEXT_06, Database root updated for all moved areas
%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 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 MOVTEXT_00, DOFULLBCK, "operation completed"
%RMU-I-MOVTEXT_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

```

3.1.6 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 SYSS\$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 SYSS\$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)
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.
$
```

3.1.7 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>
```

3.1.8 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>

```

3.1.9 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)

```

3.1.10 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 SYS\$OUTPUT.
- **NOECHO**
If the option NOECHO is used, output to SYS\$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.

3.1.11 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>
```

3.1.12 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>
```

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.

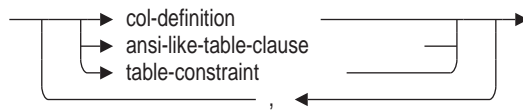
The syntax for a similar feature in the ANSI/ISO SQL Database Language moves the LIKE clause into the section that defines the columns and constraint. This adds the ability to copy column definitions from more than one table, control how GENERATED, AUTOMATIC, IDENTITY and COMPUTED columns are inherited, 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>
```

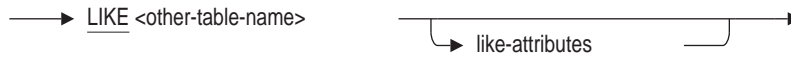
By default, GENERATED, AUTOMATIC, IDENTITY and COMPUTED columns are not copied but columns representing the same data types are created instead.

Syntax

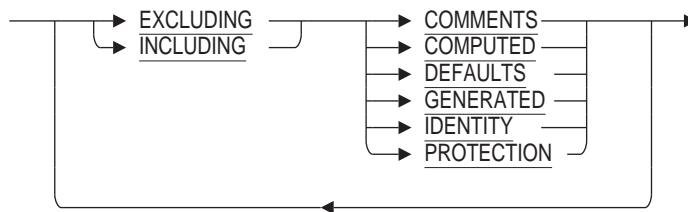
column-constraint-list =



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>

```

3.1.13 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_yminterval (: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_YMINTERVAL ( 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_YMINTERVAL**, **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_YMINTERVAL` 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.

[+ | -] PnYnMnDnTnHnMnS

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 6 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.

3.1.14 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

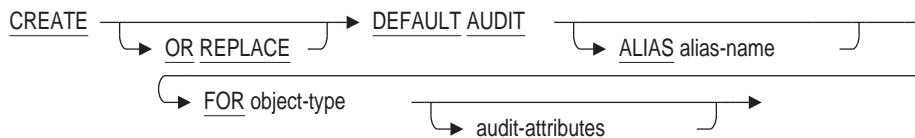
See also ALTER DEFAULT AUDIT Statement and the DROP DEFAULT AUDIT Statement.

Environment

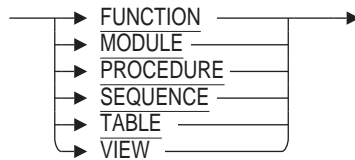
You can use the CREATE DEFAULT AUDIT statement:

- 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

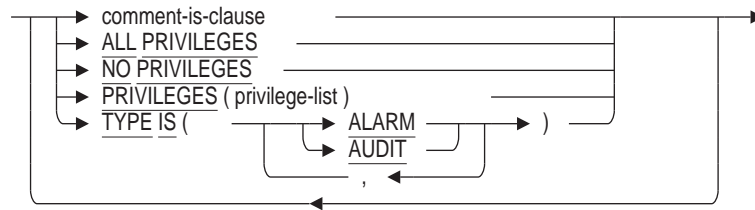
Syntax



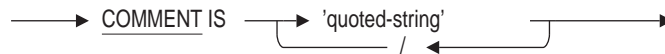
object-type =



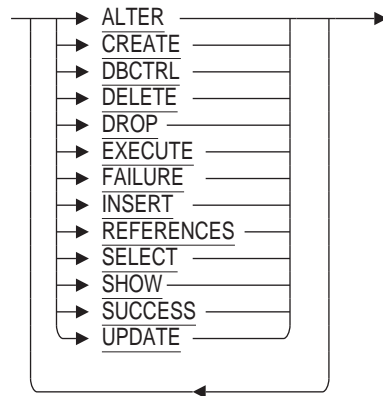
audit-attributes =



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.

3.1.15 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

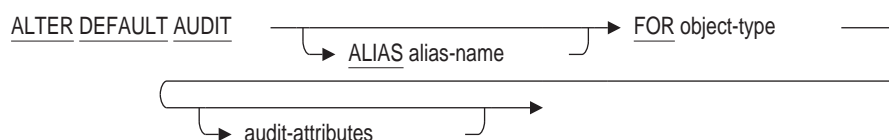
See also CREATE DEFAULT AUDIT Statement and the DROP DEFAULT AUDIT Statement.

Environment

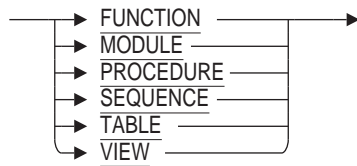
You can use the ALTER DEFAULT AUDIT statement:

- 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

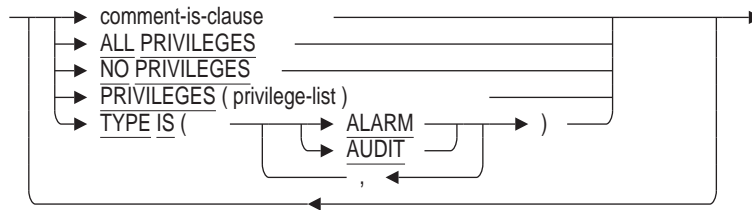
Syntax



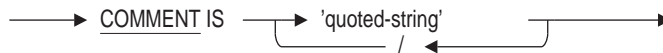
object-type =



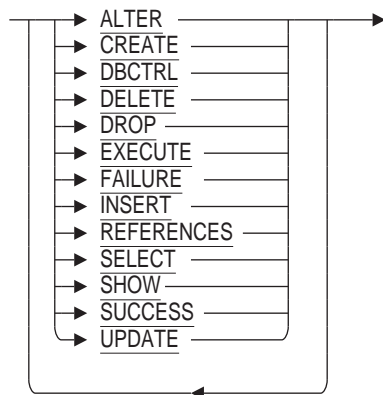
audit-attributes =



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.

3.1.16 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

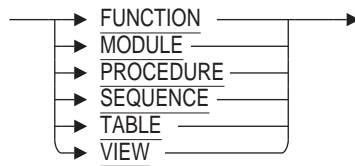
You can use the DROP DEFAULT AUDIT statement:

- 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

DROP DEFAULT AUDIT 

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.

3.1.17 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>

3.1.18 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) qualfier 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 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, -
  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

```



```

%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)
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
%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 reopen count 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
%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 WorkAlg: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)
$
```

Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.0

4.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.0

4.1.1 Intel Itanium Processor 9700 “Kittson” Certified

For this release of Oracle Rdb on HPE Integrity servers, the Intel Itanium Processor 9700 series (i6), 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.

4.1.2 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>

```

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

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..... 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	1	0	0.5	222	74.0
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

Node: TSTNOD (1/1/16) Oracle Rdb V7.3-300 Perf. Monitor
 Rate: 3.00 Seconds Logical Area Statistics
 Page: 1 of 1 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

```

4.1.4 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;
$

```

4.1.5 RMU/RESTORE Now Always Displays the %RMU-I-AIJREFUL Message

Bug 14375975

With this release of Oracle Rdb, the RMU message RMU-I-AIJREFUL 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 AIJREFUL 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-AIJREFUL 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/NOCCD/NORECOVER/NOLOG ABC_SAVE.RBF
%RMU-I-AIJREFUL, 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=O, 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

```

4.1.6 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

```

4.1.6.1 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, ' '), ' ')

4.1.6.2 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 4-1 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>
```

(continued on next page)

Example 4–1 (Cont.) Using the FIRST_VALUE Function

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 4–2 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 4–3 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 4–4 Using the GROUP_CONCAT Function

(continued on next page)

Example 4-4 (Cont.) 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>
```

4.1.7 -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

```

4.1.8 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
.
.
.

```


4.1.9 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.

%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 unjournalized modifications were made to the database, or a copy of the database if the /ROOT qualifier was specified, that were not journaled. 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 journaled 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.
$ 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.ajj;
%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.ajj
%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

```

4.1.10 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.
$
```

4.1.11 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.

```

4.1.12 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

```

4.1.13 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 =
-----+-----+-----+-----+-----+-----+-----+
|                                     |> CHECK ( predicate ) ---+
|                                     |
+---> 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.

4.1.14 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>

```

4.1.15 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
  !...
SQL>

```

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.

4.1.16 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

```

4.1.17 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.

4.1.18 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 `NONNULL_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];
    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.

4.1.19 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
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;

```

- 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;
```

4.1.20 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 SYSSOUTPUT (which cannot be used by /EXECUTE).
- Prefix=test-string
NoPrefix
By default, the generated symbol names start with RMUS. 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 4–5 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)'
```


4.1.21 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
```

4.1.22 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:

```

Enhancements And Changes Provided in Oracle Rdb Release 7.3.2.1

5.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.2.1

5.1.1 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.

5.1.2 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 "RDMSAIJ_", 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 'RDMSAIJ_' 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;
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$aij*
RDM$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"
```

5.1.3 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
```

5.1.4 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>
```

5.1.5 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

```

5.1.6 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.

5.1.7 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:

```
- 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
```

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 5–1 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 5–2 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
- 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
```

5.1.8 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.

```
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.

5.1.9 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.

5.1.10 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"
```

- 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
$ SHOW SYMBOL RDM$LOGMINER_STATE
  RDM$LOGMINER_STATE == "ENABLED"
$!
$ SQL
  DROP DATABASE FILE DEVICE:[DIRECTORY]MF_PERSONNEL;
EXIT
$ DELETE *.AIJ;*
$ RMU/RESTORE/NOCD/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
  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 -
    ! 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"

```

5.1.11 New /**[NO]MBX_ASYNCH** 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.

5.1.12 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.

5.1.13 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
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>
```

Enhancements And Changes Provided in Oracle Rdb Release 7.3.2.0

6.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.2.0

6.1.1 New COMPRESSION OCTETS Clause for CREATE INDEX Statement

This release of Oracle Rdb allows user specification of the octets used for run length compression.

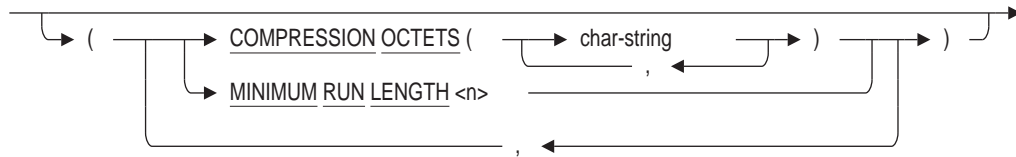
The COMPRESSION OCTETS clause can be used to specify those characters that will be used for run-length compression. If this clause is not defined, then Oracle Rdb will use the space character of the character set of any CHAR, or VARCHAR columns in the index, and a zero for any other data type.

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'`.

6.1.2 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.

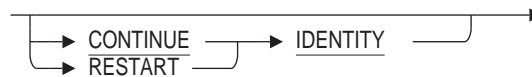
By default, the TRUNCATE TABLE statement restarts the associated IDENTITY column using the START WITH value of the sequence. There are now two new clauses to control the action of TRUNCATE TABLE when an IDENTITY column exists. Either the RESTART IDENTITY action or CONTINUE IDENTITY actions can be specified. 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.

6.1.3 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;
$
```

6.1.4 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.

6.1.5 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.

6.1.6 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 SYSSSET_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.

6.1.7 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 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 'NOREWRITE(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.

6.1.8 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>

```

6.1.9 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
%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
$

```

6.1.10 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.

Example 6–1 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
```

This example shows the PROCESS_GLOBAL option as well as an indication from RMU/SHOW USER that it has been specified.

6.1.11 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 SYSS\$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?

6.1.12 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 6.1.12.2, Per-Process Monitoring Facility Activation for activation details.

6.1.12.1 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.

6.1.12.2 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 LNMS\$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 LNM\$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 6.1.12.4, 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
RDM\$BIND_GLOBAL_STATISTICS	LNMS\$FILE_DEV	Application processes
RDM\$BIND_ABS_GLOBAL_STATISTICS	LNMS\$SYSTEM_TABLE	AIJ Backup Server
RDM\$BIND_ALS_GLOBAL_STATISTICS	LNMS\$SYSTEM_TABLE	AIJ Log Server
RDM\$BIND_LCS_GLOBAL_STATISTICS	LNMS\$SYSTEM_TABLE	Log Catch-up Server
RDM\$BIND_LRS_GLOBAL_STATISTICS	LNMS\$SYSTEM_TABLE	Log Replication Server
RDM\$BIND_RCS_GLOBAL_STATISTICS	LNMS\$SYSTEM_TABLE	Row Cache Server

6.1.12.3 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 6.1.12.4, 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
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.

6.1.12.4 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 RDMS\$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.

6.1.12.5 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.

6.1.12.6 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 AsyncWrites
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.

Process.ID	Objs.Shrd	Objs.Excl	Objs.Rfrsh	Objs.Updt	Objs.Write	Objs.Relsd
2A46BE1F:1s	135255	25	52	24	24	135280
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
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	Rec.Fetched	Rec.Marked	Rec.Stored	Pag.Checkd	Pag.Discard	Rec.Erased
2A46BE1F:1s	0	0	0	0	0	0
2A562A21:1s	0	0	0	0	0	0
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.

* **Rec.Fetched**

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.

* **PagDiscard**

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	SnprRtrv	SnplnFtch	SnppagRead	SnprRstor	SnppagFull	SnplckCnft
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.

* **SnprRtrv**

This field gives the number of records retrieved by read-only transactions.

* **SnplnFtch**

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.

* **SnprRstor**

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.

6.1.13 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 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-INVBACFIL 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-INVBACFIL, 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, 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.
```

6.1.14 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.

6.1.15 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
$
```


6.1.16 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.

6.1.17 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"
  - 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"

```


6.1.18 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>
```

6.1.19 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.

6.1.20 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 SYSSACMW 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 SYSSACMW) 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.

6.1.21 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 6–2 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 6–3 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
$
```

6.1.22 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
INSERT INTO TABLE1 VALUES (1, 'Text');
1 row inserted
COMMIT;
EXIT;
$
$ ! Switch to the next defined journal.
$
$ 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;
$

```

6.1.23 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.

6.1.24 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.

6.1.25 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 TCPIP
```

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.

6.1.26 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.

Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.3

7.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.3

7.1.1 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.

Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.2

8.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.2

8.1.1 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
```

8.1.2 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%
** 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.

8.1.3 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
$!
$! 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/NOCD/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-AIJSUCCE, 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

```

8.1.4 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.

8.1.5 RMU Extract Now Extracts SYS_GET_DIAGNOSTIC Function

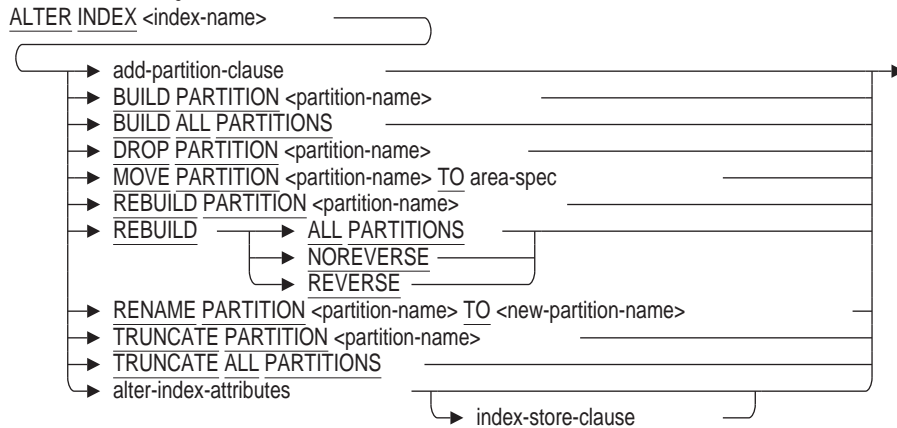
With this release of Oracle Rdb, the RMU Extract command now correctly formats the SYS_GET_DIAGNOSTIC function, which was added in earlier releases.

8.1.6 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.

8.1.7 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.

```

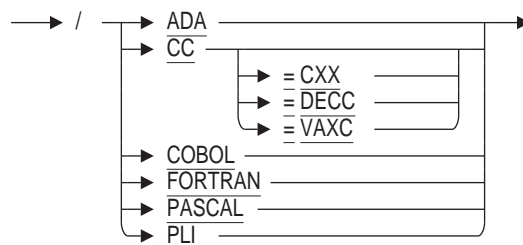
#ifdef __cplusplus
extern "C" {
#endif /* __cplusplus */
...
#ifdef __cplusplus
}
#endif /* __cplusplus */

```

- `SQLCODE` is expected to be defined as type `int`.
- On OpenVMS Alpha systems, the application is expected to be linked with the special library `SYSSLIBRARY:LIBCXXSTD.OLB`. Please refer to the relevant HPE C++ documentation.

Syntax

The revised syntax for the `PRE-LANG-QUALIFIERS` qualifier is:
`pre-lang-qualifiers =`



8.1.8 New RMU/RECOVER Feature to Alter Storage Areas, After_Journal and Record_Cache Directories

Enhancement Bug 867636

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:

```
/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)
```

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]RECOVPTAIJ.OPT, -  
ROW_CACHE=DISK:[DIRECTORY]RECOVPTRCACHE.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.

```

! 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 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.
$
$ 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/NOCCD/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-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-AIJSUCCES, database recovery completed successfully

```

```

%RMU-I-AIJFNLSEQ, 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/NOCCD/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
$
$ 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-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-AIJSUCCE, 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

```

8.1.9 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 */
$

```

8.1.10 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-SOFTREERRS, 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-SOFTREERRS, 5 recoverable media errors occurred reading
DEVICE:[DIRECTORY]MF_PERSONNEL.RBF;
$ SHOW SYMBOL $STATUS
  $STATUS == "%X12C8821A"

```

8.1.11 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.

Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.1

9.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.1

9.1.1 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.

9.1.2 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.

9.1.3 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.

9.1.4 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
%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-BSGPGGLARE, 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-ESGPGGLARE, 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
%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"
$
```

Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.0

10.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.0

10.1.1 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.

10.1.2 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
```

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.

```
$ 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
```

10.1.3 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 retrieval
%RMU-I-BGNAIPVER, beginning AIP pages verification
%RMU-I-ENDAIPVER, completed AIP pages verification
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-I-ENDABMSPM, completed ABM pages verification
%RMU-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
$

```

10.1.4 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.

10.1.5 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.

10.1.6 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.

10.1.7 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.

10.1.8 /[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.

```
$ 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.

```

10.1.9 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

```

10.1.10 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.

10.1.11 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.

10.1.12 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.

10.1.13 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
```


10.1.14 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.

10.1.15 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/NOCD/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> 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/NOCD/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/NOCD/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/NOCD/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/NOCD/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/NOCD/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
```

10.1.16 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
```

10.1.17 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
```

10.1.18 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.

10.1.19 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.

10.1.20 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>
```

10.1.21 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.

10.1.22 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.

10.1.23 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.

```

$ 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          BADATA = 00
ROOT = 01              AREA_ID = 0          LAREA_ID = 0
PNO = 0

00000000000000000000000000000000A002006AC 0000  ' . . . . . '

KODA database root record
0000000000000049544F4F52534D4452 0000  'RDMSROOTI.....'
2EC700C900A66EB9EBCF255B00000000 0010  '.....[%Iè'n.É.Ç.'
0000002100000088000000170000000F 0020  '.....!....'
0000000B000000280000001900000022 0030  '".....(.....'
00000070000000800000003A0000006C 0040  'l.....p...'
0000001000000040000001200000001 0050  '.....'
00000014000000140000001000000032 0060  '2.....'
0000000300000005000000FA00000006 0070  '....ú.....'
00000000000000A0000000A0000000A 0080  '.....'
00000000000000000000000000000000 0090  '.....'
00000100000000000000000000000000 00A0  '.....'
00000000000000000000000000000000 00B0  '.....'
::: (1 duplicate line)
00000003000000020000000200000000 00D0  '.....'
000000080000000800AC93EB3CA7391C 00E0  '.9§<ë.....'
00000001000000040000000500000005 00F0  '.....'
00000000000000040000002400000002 0100  '....$. . . . . '
00000004000000010000008A00000000 0110  '.....'
000000000000000000000000FF00000004 0120  '.....'
00000200000000000000000000000000 0130  '.....'
20571AEA00000000000000200000008B 0140  '.... . . . . ê.W '
0000000000AC93EB216424E900AC93EB 0150  'è...é$d!è.....'
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'
00000000000000000000000000000000 0340 'EL.RDB;1.....'
00000000000000000000000000000000 0350 '.....'
                                :::: (52 duplicate lines)
                                000000000000000000000000 06A0 '.....'

REC_SIZE = 1706          REC_TYPE = 33          BADDATA = 00
ROOT = 01              AREA_ID = 0             LAREA_ID = 0
PNO = 0

00000000000000000000000000000000A002106AA 0000 'a.!.....'

KODA database root record
00000000000000000000000000000000 0000 '.....'
                                :::: (105 duplicate lines)
                                000000000000000000000000 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
00000000000000000000000000000000 0000 '.....'
                                :::: (105 duplicate lines)
                                000000000000000000000000 06A0 '.....'

```

10.1.24 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>

```

10.1.25 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 SYSSMANAGER: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=SYSSOUTPUT
/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 .AUDITSJOURNAL. 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 SYSS\$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 10–1 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	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$K_PKT_PRIV_DESIRED  :  DBCTRL
RDBNSA$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.

```

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>_${DGA174}:[PRODUCTION.DATABASES]EVENTS_DB.RDB;1</database_name>
<audit_name>SECURITY</audit_name>
  <system_id>44390</system_id>
  <image_name>DISK$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$K_PKT_DBNAME   : _${DGA174}:[PRODUCTION.DATABASES]EVENTS_DB.RDB;1
NSA$C_PKT_AUDIT_NAME  : SECURITY
NSA$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.
$

```


10.1.26 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           0           0           0
           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>
```

10.1.27 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
```

10.1.28 /[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

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

```

10.1.29 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.

10.1.30 /[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.

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

```

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
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.

```

10.1.31 /[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

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

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

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

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.

```

10.1.32 /[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.

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
```

```
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)	

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

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)

```

MIN IO path length vs. frequency

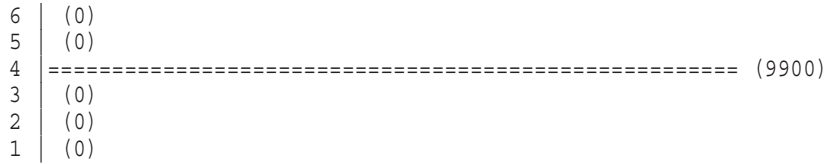
```

6 | (0)
5 | (0)
4 | (0)
3 | ===== (83)
2 | ===== (12)
1 | (0)

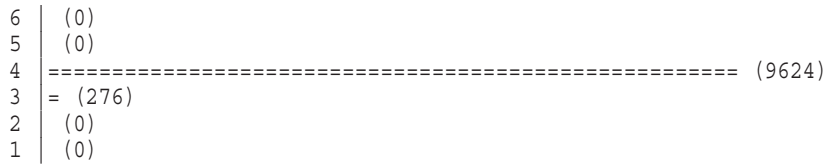
```

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

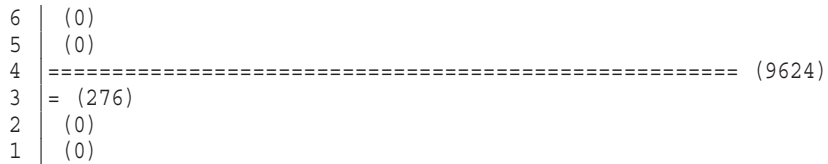
dbkey path length vs. frequency



MAX IO path length vs. frequency



MIN IO path length vs. frequency



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.

```

10.1.33 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 SYS\$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).

10.1.34 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' "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"

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

```

10.1.35 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.

10.1.36 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>
```

10.1.37 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.

10.1.38 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.

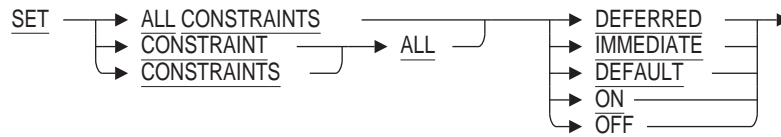
In addition to these changes, the /DACCESS=TABLE option shows that the audited relation is a view or a table.

Note

At this time, RMU/SHOW AUDIT accesses multischema databases using MULTISHEMA IS OFF. Therefore, the external (possibly generated) names will be displayed for all objects.

10.1.39 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.

10.1.40 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.

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>
```

10.1.41 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>
```

10.1.42 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.

10.1.42.1 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>
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>

```

10.1.42.2 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 =

→ **RELEASE SAVEPOINT** alias-name . savepoint-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 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>
```

10.1.42.3 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 `RDBS_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>

```

10.1.43 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.

10.1.44 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 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
*
* -----
```



```

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$1$DGA220846900100000000000001"
    - 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)
  - 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
*
* 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...
- 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

```

10.1.45 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

10.1.46 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 10–1 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 10–2 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};
...
```

10.1.47 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.

```
$      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
```

10.1.48 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>
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;

```


10.1.49 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.
$
```

10.1.50 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 10–3 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 10–4 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>
```

10.1.51 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 10-5 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>
cont> end module;
```

10.1.52 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 SYSS\$SHARE:SQL_LITERALS.H.

```

#define SQLDA_VARBINARY 909
#define SQLDA_BINARY 913

```

10.1.53 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$dbhandle
Default alias:
  Oracle Rdb database in file TESTING_DB
  Multischema mode is disabled
.
.
.
  Shared Memory:          Process
  Large Memory:           Disabled
Security Checking is External
  System Index Compression is ENABLED
  System Index:
    Type is sorted ranked
    Prefix cardinality collection is enabled
  Logminer support is disabled
  Galaxy support is disabled
  Prestarted transactions are enabled
  Dictionary Not Required
  ACL based protections
Storage Areas in database with filename TESTING_DB
  RDB$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>

```

10.1.54 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.

10.1.55 New WITH Clause Provides Subquery Factoring

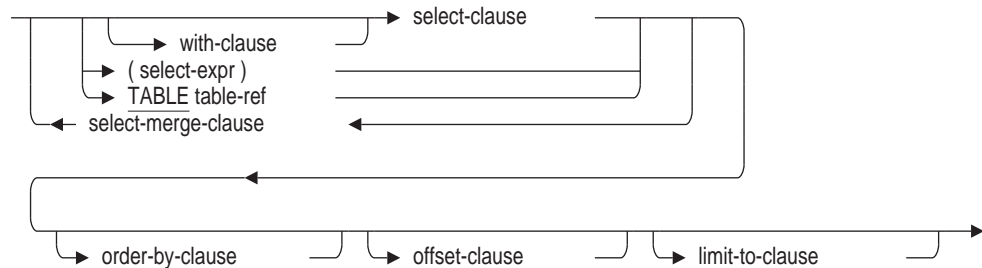
This release of Oracle Rdb introduces the WITH clause as part of the SELECT expression. This feature, known as subquery factoring, allows the SQL programmer to simplify complex queries by creating named subqueries that can be used (possibly multiple times) in the associated SELECT expression.

The following example shows the declaration of two subquery factors, EMP and DPT, that are used in the select expression.

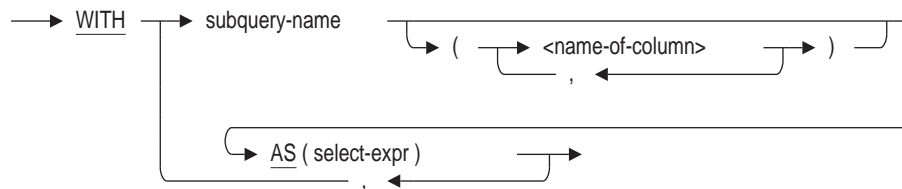
```
SQL> with emp as (select *
cont>                from employees inner join
cont>                job_history using (employee_id)
cont>                where job_end is null),
cont> dpt as (select * from departments)
cont> select e.last_name, d.department_name, m.last_name as MANAGER
cont> from emp e
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
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
      x
%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 10–6 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 10–7 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 10–8 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>
```

10.1.56 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.

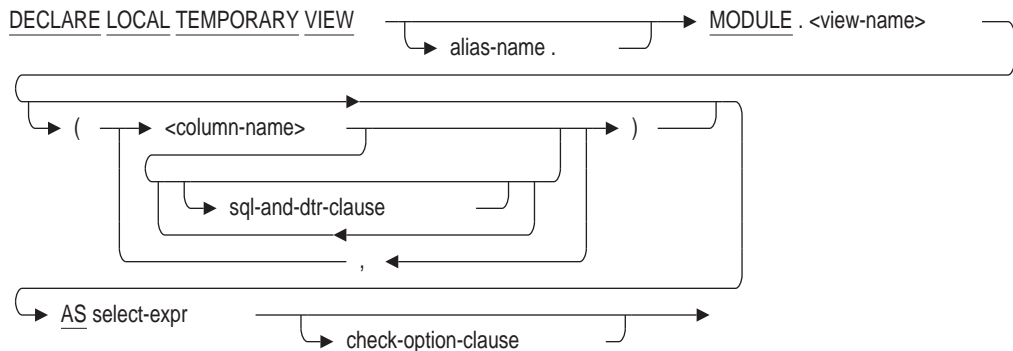
This statement allows an application to define view definitions that are temporary and do not require CREATE privilege on the database.

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.

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.

10.1.57 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.

Example 10–9 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';
              When
EID          NUM_JOBS  Started          CURRENT_START
00 164             2    5-JUL-1980     21-SEP-1981
1 row selected
SQL>
```

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>
```

10.1.58 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.

Example 10–10 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;

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 `RDMSS$ENABLE_BITMAPPED_SCAN` as 1, `RDMSS$SET_FLAGS` as "BITMAPPED_SCAN", or use the `SET_FLAGS 'BITMAPPED_SCAN'` statement in the application.

10.1.59 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.

10.1.60 New SQL Functions Added

This release of Oracle Rdb adds new functions to the SYSS\$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.

10.1.61 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.

10.1.62 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.
$
```

10.1.63 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.

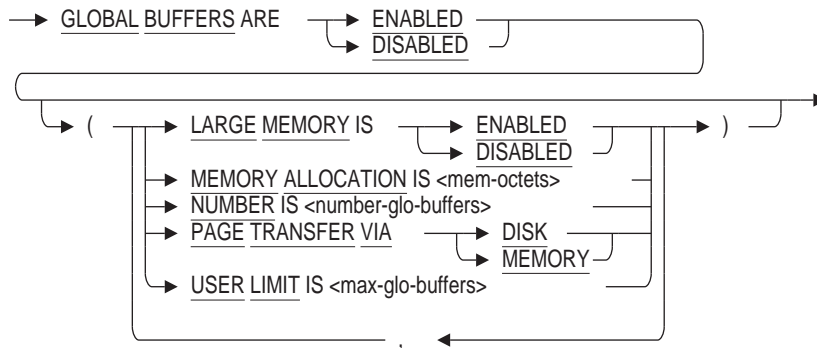
```
create database
  filename TEST_DB

  allocation 110k pages
  snapshot allocation 1k pages
  global buffers are disabled (memory allocation 1m)

  create storage area AREA1
    allocation 100k pages
    snapshot allocation 2k pages
;
```

Syntax

global-buffer-params=



10.1.64 New REPLACE Statement

Bug 8929218

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.

10.1.65 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 'NOREWRITE(IN_CLAUSE)' and re-enabled using SET FLAGS 'REWRITE(IN_CLAUSE)'.

This new feature was introduced in Oracle Rdb Release 7.3.1.0.

Optimizer Enhancements Appendix

11.1 Optimizer Enhancements

11.1.1 Changes and Improvements to the Rdb Optimizer and Query Compiler

These changes were made available in Oracle Rdb Release 7.3.1.0.

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 contains 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