Oracle® Rdb for OpenVMS

Table of Contents

Oracle® Rdb for OpenVMS....1

Release Notes....2

May 2010....3

Contents....4

Preface....5

Purpose of This Manual....6

Intended Audience....7

Document Structure....8

Chapter 1Installing Oracle Rdb Release 7.2.4.1....9

1.1 Oracle Rdb on HP OpenVMS Industry Standard 64....10

1.2 Requirements....11
- 1.2.1 Ensure No Processes Have RDMSHRP Image Activated....11

1.3 Intel Itanium Processor 9100 "Montvale" Support....13

1.4 Maximum OpenVMS Version Check....14

1.5 Database Format Changed....15

1.6 Using Databases from Releases Earlier than V7.0....16

1.7 Invoking the VMSINSTAL Procedure....17

1.8 Stopping the Installation....18

1.9 After Installing Oracle Rdb....19

1.10 VMS$MEM_RESIDENT_USER Rights Identifier Required....20

1.11 Installation, Configuration, Migration, Upgrade Suggestions....21

Chapter 2Software Errors Fixed in Oracle Rdb Release 7.2.4.1....24

2.1 Software Errors Fixed That Apply to All Interfaces....25
- 2.1.1 Process Termination Due to Register Stack Engine (RSE) Stack Overflow....25
- 2.1.2 Query Using Multiple IN Clauses With DECODE Function Bugchecks....25
- 2.1.3 Intermittent RDB−E−NO_DUP Index Field Value Already Exists....26
- 2.1.4 Query Using Bitmap Bugchecks....26

Table of Contents

2.1 Software Errors Fixed That Apply to All Interfaces
- 2.1.5 Wrong Result From Query With Match Strategy....26
- 2.1.6 Unexpected Error After Truncating a Row Cached Table....28
- 2.1.7 Wrong Result From Aggregate Match Join With Distinct....29
- 2.1.8 Using RDB Remote Created a Logfile Called RDBSERVER.EXE....31
- 2.1.9 Wrong Result From LIMIT TO Query With ORDER BY DESC....32
- 2.1.10 Wrong Result From OR Predicate With Constants as Index Boolean....34

2.2 SQL Errors Fixed....37
- 2.2.1 Not all Errors Were Written to Log File Created by SET OUTPUT....37
- 2.2.2 Cardinality Changes Lost by ALTER STORAGE MAP....37
- 2.2.3 CHECK Constraint for Declared Variables Not Supported on Integrity Systems....38
- 2.2.4 Unexpected Bugcheck When Dropping a LIST OF BYTE VARYING Column....38
- 2.2.5 Unexpected SQL−E−TRUN−STORE Error When Using UPPER or LOWER Function....39
- 2.2.6 Unexpected INVALID_BLR Error From ALTER TABLE ... ALTER COLUMN....39
- 2.2.7 Unexpected Bugcheck Reporting RDMS−E−SIGNATURE_MISMA, Invalid Parameter Signature on Procedure Call....40

2.3 RDO and RDML Errors Fixed....41
- 2.3.1 RDO IMPORT Alignment Faults....41

2.4 RMU Errors Fixed....42
- 2.4.1 RMU/RECOVER/ORDER_AIJ_FILES Fails With RMS−E−FNF When Processing One After Image Journal File....42

2.5 Row Cache Errors Fixed....43
- 2.5.1 RMU /POPULATE_CACHE Not Correctly Fetching System Records....43

Chapter 3Software Errors Fixed in Oracle Rdb Release 7.2.4.0....44

3.1 Software Errors Fixed That Apply to All Interfaces....45
- 3.1.1 Bugcheck With SYSTEM−F−ROPRAND in KODTXN$POST_TSNBLK_UPDATE....45
- 3.1.2 Memory Leak On Systems With RAD Support Enabled....46
- 3.1.3 Incorrect Messages From RMU /MONITOR START When RDM$MON_USERNAME Specifies Non−existant Account....46
- 3.1.4 Potential INVEXCEPTN System Crash Using LARGE MEMORY IS ENABLED or SHARED MEMORY IS SYSTEM on Itanium....46
- 3.1.5 Wrong Result From Query With Derived Table....47
- 3.1.6 Application Looses Virtual Memory (Memory Leak)....53
- 3.1.7 VMS$BUFFER_OBJECT_USER Not Always Checked....53
- 3.1.8 Deleted Space in Uniform Areas Not Reclaimed by Other Users When Erased Rows Moved From Row Cache to Disk....53
- 3.1.9 LOWER Function Problem When Using ISOLATINCYRILLIC Character Set....54
- 3.1.10 Query Bugchecks with SYSTEM−F−FLTINV....54
- 3.1.11 Bugcheck Altering Storage Map....55
- 3.1.12 Wrong Results when Expression Shared Between Aggregate Filters....56
- 3.1.13 Optimize For Sequential Access Using Index....56
- 3.1.14 Query With Match Strategy for Left Outer Join Bugchecks....57

Table of Contents

3.1 Software Errors Fixed That Apply to All Interfaces
3.1.15 Bugchecks at DIOFETCH$FETCH_SNAP_SEG or DIOCCH$FETCH_SNAP_SEG............57
3.1.16 Bugcheck in PSII2INSERTDUPBBC..58

3.2 SQL Errors Fixed...59
3.2.1 Unexpected Failures of TRUNC and ROUND Functions..59
3.2.2 Unexpected Restrictions in Date/Time Subtraction..59
3.2.3 Unexpected DEFVALINC Error When Altering a Column's Domain.........................60
3.2.4 Unexpected FLDNOTCRS Error When Referencing DBKEY or ROWID in Quotes...............60
3.2.5 Unexpected Bugcheck When Assigning Value to FOR Cursor Variables...................61
3.2.6 Unexpected Bugcheck When Executing External Routine.......................................62
3.2.7 Unexpected Error When Importing a Database With Profile Definitions....................62
3.2.8 Unexpected Bugcheck When Table Partitions Used in RESERVING Clause................63
3.2.9 Unexpected Failure of GRANT and REVOKE When Using Synonyms........................64
3.2.10 Unexpected Correlation Names Appearing as Column Headers..............................64
3.2.11 Wrong Results When CONCAT Used in an Aggregate Expression with DISTINCT Clause..65
3.2.12 UPDATE ONLY CURSOR Clause Quietly Ignored by SQL Module Language Processor and SQL Precompiler...65

3.3 RMU Errors Fixed...66
3.3.1 Application Hangs Using Automatic AIJ Backups...66
3.3.2 Failed RMU MOVE or COPY Deletes Wrong Files..66
3.3.3 RMU Extract Now Extracts Comments for Constraints...66
3.3.4 Bugcheck in RMU /COLLECT OPTIMIZER_STATISTICS /STATISTICS=WORKLOAD...67
3.3.5 Increased Cardinality Limit in RMU /COLLECT OPTIMIZER_STATISTICS /STATISTICS=WORKLOAD...67
3.3.6 Incorrect LOCK_TIMEOUT With RMU /BACKUP /ONLINE /LOCK_TIMEOUT /LOG......68
3.3.7 RMU /RESTORE /ONLINE /JUST_CORRUPT Bugcheck...69
3.3.8 RMU /LOAD /PARALLEL COSI−F−READERR Without VMS BYPASS Privilege.............70

3.4 LogMiner Errors Fixed..71
3.4.1 Continuous LogMiner Startup Serialized With AIJ Backups....................................71
3.4.2 RMU /UNLOAD /AFTER_JOURNAL Allows /RESTART Without /CONTINUOUS............71

3.5 Row Cache Errors Fixed...72
3.5.1 Incremental Backup With Row Cache...72

3.6 Hot Standby Errors Fixed..73
3.6.1 Hot Standby LRS Database Prefetch Count Limit...73

Chapter 4Enhancements And Changes Provided in Oracle Rdb Release 7.2.4.1....................74

4.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.4.1...........................75
4.1.1 New SET LOGFILE Command..75
4.1.2 SET FLAGS Statement Now Allows ON ALIAS Clause...76
4.1.3 SQL Compiler−Generated Name Uniqueness Enhanced..77
4.1.4 Reduced CPU Usage and Improved Performance...77
4.1.5 New RMU /SHOW STATISTICS /WRITE_REPORT_DELAY=n Feature.......................78

Table of Contents

Chapter 5Enhancements And Changes Provided in Oracle Rdb Release 7.2.4........79

5.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.4........80
5.1.1 Date/Time Arithmetic Enhancements........80
5.1.2 New DEFAULT PROFILE Feature........80
5.1.3 RMU /DUMP/BACKUP/OPTIONS=ROOT /HEADER_ONLY Displays the Header Information Only........83
5.1.4 GET ENVIRONMENT Now Supports SQLCODE and SQLSTATE Capture........85
5.1.5 Timestamp Added to Messages For RMU LOAD and UNLOAD........86
5.1.6 New SET SQLDA Statement........86
5.1.7 RMU /SHOW VERSION Displays System Architecture and Version........90
5.1.8 New IDENT Option for SQL Module Language PRAGMA Clause........90
5.1.9 New Keyword for SQL Module Language /PRAGMA Qualifier........93
5.1.10 New IDENT Option for SQL Precompiler DECLARE MODULE Statement........93
5.1.11 New Keyword for SQL Precompiler PRAGMA Option........95
5.1.12 RDB_STATS_DATABASE Example Program........95
5.1.13 RCS Time−Based Cache Sweeping........97
5.1.14 RMU Command TSN Keyword and Qualifier Value........98
5.1.15 New Support for RENAME and CREATE SYNONYM Commands........98

Chapter 6Documentation Corrections, Additions and Changes........101

6.1 Documentation Corrections........102
6.1.1 RMU /SET ROW_CACHE Command Updates........102
6.1.2 Documentation for the DEBUG_OPTIONS Qualifier of RMU Unload........103
6.1.3 SQL$MSGxx.DOC Is Not Alphabetical........104
6.1.4 LOCK_TIMEOUT Documentation Error in RMU Reference Manual Release 7.2........105
6.1.5 Revised Example for SET OPTIMIZATION LEVEL Statement........105
6.1.6 RMU /VERIFY Process Quotas and Limits Clarification........106
6.1.7 Online Backup Can Be Performed With Transfer Via Memory........107
6.1.8 Missing Example for CREATE STORAGE MAP........107
6.1.9 RDM$BIND_MAX_DBR_COUNT Documentation Clarification........109
6.1.10 Database Server Process Priority Clarification........110
6.1.11 Explanation of SQL$INT in a SQL Multiversion Environment and How to Redefine SQL$INT........111
6.1.12 Clarification of PREPARE Statement Behavior........112
6.1.13 RDM$BIND_LOCK_TIMEOUT_INTERVAL Overrides the Database Parameter........112
6.1.14 Missing Tables Descriptions for the RDBEXPERT Collection Class........113
6.1.15 Missing Columns Descriptions for Tables in the Formatted Database........114

6.2 Address and Phone Number Correction for Documentation........122

6.3 Online Document Format and Ordering Information........123

Chapter 7Known Problems and Restrictions........124

Table of Contents

7.1 Known Problems and Restrictions in All Interfaces....125
7.1.1 Possible Incorrect Results When Using Partitioned Descending Indexes....125
7.1.2 Remote Attach Stalls Before Detecting a Node is Unreachable....126
7.1.3 Case Sensitive Values in RDB$CLIENT_DEFAULTS.DAT....127
7.1.4 Standalone WITH Clause in Compound Statements Now Deprecated....127
7.1.5 Calling DECC$CRTL_INIT....128
7.1.6 Application and Oracle Rdb Both Using SYS$HIBER....128
7.1.7 Unexpected RCS Termination....130
7.1.8 Possible Incorrect Results When Using Partitioned Descending Indexes on I64....130
7.1.9 Changes for Processing Existence Logical Names....131
7.1.10 Patch Required When Using VMS V8.3 and Dedicated CPU Lock Manager....132
7.1.11 SQL Module or Program Fails with %SQL−F−IGNCASE_BAD....132
7.1.12 External Routine Images Linked with PTHREAD$RTL....133
7.1.13 Using Databases from Releases Earlier than V7.0....133
7.1.14 Partitioned Index with Descending Column and Collating Sequence....133
7.1.15 Domain−Qualified TCP/IP Node Names in Distributed Transactions....134
7.1.16 ILINK−E−INVOVRINI Error on I64....136
7.1.17 New Attributes Saved by RMU/LOAD Incompatible With Prior Versions....136
7.1.18 SYSTEM−F−INSFMEM Fatal Error With SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED in Galaxy Environment....136
7.1.19 Oracle Rdb and OpenVMS ODS−5 Volumes....137
7.1.20 Optimization of Check Constraints....138
7.1.21 Carryover Locks and NOWAIT Transaction Clarification....140
7.1.22 Unexpected Results Occur During Read−Only Transactions on a Hot Standby Database....140
7.1.23 Row Cache Not Allowed While Hot Standby Replication is Active....141
7.1.24 Excessive Process Page Faults and Other Performance Considerations During Oracle Rdb Sorts....141
7.1.25 Control of Sort Work Memory Allocation....142
7.1.26 The Halloween Problem....143

7.2 SQL Known Problems and Restrictions....145
7.2.1 SET FLAGS CRONO_FLAG Removed....145
7.2.2 Interchange File (RBR) Created by Oracle Rdb Release 7.2 Not Compatible With Previous Releases....145
7.2.3 Single Statement LOCK TABLE is Not Supported for SQL Module Language and SQL Precompiler....145
7.2.4 Multistatement or Stored Procedures May Cause Hangs....146
7.2.5 Use of Oracle Rdb from Shareable Images....147

7.3 Oracle RMU Known Problems and Restrictions....148
7.3.1 RMU Convert Fails When Maximum Relation ID is Exceeded....148
7.3.2 RMU Unload /After_Journal Requires Accurate AIP Logical Area Information....148
7.3.3 Do Not Use HYPERSORT with RMU Optimize After_Journal Command....149
7.3.4 Changes in EXCLUDE and INCLUDE Qualifiers for RMU Backup....150
7.3.5 RMU Backup Operations Should Use Only One Type of Tape Drive....150
7.3.6 RMU/VERIFY Reports PGSPAMENT or PGSPMCLST Errors....151

Table of Contents

7.4 Known Problems and Restrictions in All Interfaces for Release 7.0 and Earlier..............................153
7.4.1 Converting Single−File Databases..............................153
7.4.2 Row Caches and Exclusive Access..............................153
7.4.3 Exclusive Access Transactions May Deadlock with RCS Process..............................153
7.4.4 Strict Partitioning May Scan Extra Partitions..............................153
7.4.5 Restriction When Adding Storage Areas with Users Attached to Database..............................154
7.4.6 Multiblock Page Writes May Require Restore Operation..............................154
7.4.7 Replication Option Copy Processes Do Not Process Database Pages Ahead of an Application..............................155

7.5 SQL Known Problems and Restrictions for Oracle Rdb Release 7.0 and Earlier..............................156
7.5.1 ARITH_EXCEPT or Incorrect Results Using LIKE IGNORE CASE..............................156
7.5.2 Different Methods of Limiting Returned Rows from Queries..............................156
7.5.3 Suggestions for Optimal Use of SHARED DATA DEFINITION Clause for Parallel Index Creation..............................157
7.5.4 Side Effect When Calling Stored Routines..............................159
7.5.5 Considerations When Using Holdable Cursors..............................160
7.5.6 AIJSERVER Privileges..............................160

Oracle® Rdb for OpenVMS

Release Notes

Release 7.2.4.1

May 2010

Oracle Rdb Release Notes, Release 7.2.4.1 for OpenVMS

Copyright © 1984, 2010 Oracle Corporation. ***All rights reserved.***

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent and other intellectual and industrial property laws. Reverse engineering, disassembly or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error−free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

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

US GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Rederal Acquisition Regulation and agnecy−specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forthe in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227−19, Commercial Computer Software − Restricted Rights (June, 1987). Oracle Corporation, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

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

Oracle is a registered trademark, and Hot Standby, LogMiner for Rdb, Oracle CDD/Repository, Oracle CODASYL DBMS, Oracle Expert, Oracle Rdb, Oracle RMU, Oracle RMUwin, Oracle SQL/Services, Oracle Trace, and Rdb7 are trademark or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

Contents

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb Release 7.2.4.1. The notes describe changed and enhanced features; upgrade and compatibility information; new and existing software problems and restrictions; and software and documentation corrections.

Intended Audience

This manual is intended for use by all Oracle Rdb users. Read this manual before you install, upgrade, or use Oracle Rdb Release 7.2.4.1.

Document Structure

This manual consists of the following chapters:

Chapter 1	Describes how to install Oracle Rdb Release 7.2.4.1.
Chapter 2	Describes problems corrected in Oracle Rdb Release 7.2.4.1.
Chapter 3	Describes problems corrected in Oracle Rdb Release 7.2.4.0.
Chapter 4	Describes enhancements introduced in Oracle Rdb Release 7.2.4.1.
Chapter 5	Describes enhancements introduced in Oracle Rdb Release 7.2.4.0.
Chapter 6	Provides information not currently available in the Oracle Rdb documentation set.
Chapter 7	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb Release 7.2.4.1.

Chapter 1
Installing Oracle Rdb Release 7.2.4.1

This software update is installed using the OpenVMS VMSINSTAL utility.

NOTE

Oracle Rdb Release 7.2 kits are full kits. There is no requirement to install any prior release of Oracle Rdb when installing new Rdb Release 7.2 kits.

1.1 Oracle Rdb on HP OpenVMS Industry Standard 64

The Oracle Rdb product family is available on the HP OpenVMS Industry Standard 64 platform and the OpenVMS AlphaServer platform. In general, the functionality present in Oracle Rdb on OpenVMS Alpha will be available in Oracle Rdb on OpenVMS Industry Standard 64. However, certain differences between the platforms may result in minor capability and functionality differences.

The database format for Oracle Rdb Release 7.2 is the same on both I64 and Alpha platforms and databases may be accessed simultaneously from both architectures in a cluster environment. Access to an Oracle Rdb Release 7.2 database from prior Rdb versions (on Alpha or VAX platforms) or from other systems on the network is available via the Oracle Rdb remote database server.

1.2 Requirements

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

- This Oracle Rdb release requires the following OpenVMS environments:
    - ♦ OpenVMS Alpha V8.2 to V8.3−x.
    - ♦ OpenVMS Industry Standard 64 V8.2−1 to V8.3−x.
- Oracle Rdb must be shutdown before you install this update kit. That is, the command file SYS$STARTUP:RMONSTOP72.COM should be executed before proceeding with this installation. If you have an OpenVMS cluster, you must shutdown the Rdb Release 7.2 monitor on all nodes in the cluster before proceeding.
- After executing RMONSTOP72.COM, no process on any system in the cluster should have any existing RDMSHRP72.EXE image activated. See Section 1.2.1 for additional information.
- The installation requires approximately 280,000 blocks for OpenVMS Alpha systems.
- The installation requires approximately 500,000 blocks for OpenVMS I64 systems.
- Oracle strongly recommends that all available OpenVMS patches are installed on all systems prior to installing Oracle Rdb. Contact your HP support representitive for more information and assistance.

1.2.1 Ensure No Processes Have RDMSHRP Image Activated

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

The following command procedure can be used on each cluster node that shares the system disk to determine if there are any processes that have activated the RDMSHRP72.EXE image. This procedure should be executed by a privileged account after RMONSTOP72 has been run. Any processes that have RDMSHRP72.EXE activated at this point should be terminated prior to starting the Rdb installation procedure.

```
$ RDB$TMP = "SYS$SCRATCH:''F$UNIQUE()'.TMP
$ DEFINE /NOLOG /USER RDB$TMP 'RDB$TMP
$ ANALYZE /SYSTEM
    SET OUTPUT RDB$TMP
    SHOW PROCESS /CHANNELS ALL
    EXIT
$ SEARCH /OUTPUT='RDB$TMP' 'RDB$TMP';−1 RDMSHRP72.EXE,"PID:"
$ SEARCH 'RDB$TMP' RDMSHRP72.EXE /WINDOW=(1,0)
$ DELETE /NOLOG 'RDB$TMP';*{text}
```

In the following example, the process 2729F16D named "FOO$SERVER" has the image RDMSHRP72.EXE activated even after RMONSTOP72.COM has been executed and this process should be terminated prior to starting the Rdb installation procedure:

```
$ @SYS$STARTUP:RMONSTOP72.COM
   .
   .
   .
$ @FIND_RDMSHRP72_PROC.COM

OpenVMS system analyzer

Process index: 016D   Name: FOO$SERVER  Extended PID: 2729F16D
 0240  7FEF4460 8384F300 $1$DGA2:[VMS$COMMON.SYSLIB]RDMSHRP72.EXE;722
$
```

1.3 Intel Itanium Processor 9100 "Montvale" Support

For this release of Oracle Rdb on HP Integrity servers, the Intel Dual−Core Itanium 2 Processor 9100 series, code named "Montvale", is the newest processor supported.

1.4 Maximum OpenVMS Version Check

OpenVMS Version 8.3−x is the maximum supported version of OpenVMS for this release of Oracle Rdb.

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

1.5 Database Format Changed

The Oracle Rdb on−disk database format has been incremented to 721. An RMU /CONVERT operation is required for databases created by or accessed by Oracle Rdb V7.0 or V7.1 to be accessed with Rdb Release 7.2.

Prior to upgrading to Oracle Rdb Release 7.2 and prior to converting an existing database to Oracle Rdb Release 7.2 format, Oracle strongly recommends that you perform a full database verification (with the "RMU /VERIFY /ALL" command) along with a full database backup (with the "RMU /BACKUP" command) to ensure a valid and protected database copy.

1.6 Using Databases from Releases Earlier than V7.0

You cannot convert or restore databases earlier than the Oracle Rdb V7.0 format directly to Oracle Rdb V7.2 format. The RMU Convert command for Oracle Rdb V7.2 supports conversions from Oracle Rdb V7.0 and V7.1 format databases only. If you have an Oracle Rdb V3.0 through V6.1 format database or database backup, you must convert it to at least Oracle Rdb V7.0 format and then convert it to Oracle Rdb V7.2 format. For example, if you have a V4.2 format database, you must convert it first to at least Oracle Rdb V7.0 format, then convert it to Oracle Rdb V7.2 format.

If you attempt to convert or restore a database that is prior to Oracle Rdb V7.0 format directly to Oracle Rdb V7.2 format, Oracle RMU generates an error.

1.7 Invoking the VMSINSTAL Procedure

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

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

- To install the Oracle Rdb for OpenVMS I64 kit that is performance targeted for I64 platforms:

  ```
  @SYS$UPDATE:VMSINSTAL RDBV72410IM device−name
  ```
- To install the Oracle Rdb for OpenVMS Alpha kit that is compiled to run on all Alpha platforms:

  ```
  @SYS$UPDATE:VMSINSTAL RDBV72410AM device−name
  ```
- To install the Oracle Rdb for OpenVMS Alpha kit that is performance targeted for Alpha EV56 and later platforms:

  ```
  @SYS$UPDATE:VMSINSTAL RDBV72411AM device−name
  ```

device−name

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

1.8 Stopping the Installation

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

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

1.9 After Installing Oracle Rdb

This update provides a new Oracle TRACE facility definition for Oracle Rdb. Any Oracle TRACE selections that reference Oracle Rdb will need to be redefined to reflect the new facility version number for the updated Oracle Rdb facility definition, "RDBVMSV7.2".

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

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

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

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

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

Note that the process executing the INSERT DEFINITION command must use the version of Oracle Rdb that matches the version used to create the Oracle TRACE administration database or the INSERT DEFINITION command will fail.

1.10 VMS$MEM_RESIDENT_USER Rights Identifier Required

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

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

1.11 Installation, Configuration, Migration, Upgrade Suggestions

Oracle Rdb Release 7.2 fully supports mixed−architecture clusters for AlphaServer systems and HP Integrity servers.

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

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

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

Table 1−1 Migration Suggestions

Case	You Wish To...	You should...
1	Add an Integrity server to an existing cluster of Alpha servers	1. Verify database(s) using RMU/VERIFY/ALL. 2. Backup database(s) using RMU/BACKUP. 3. Install Rdb 7.2 on Integrity and Alpha nodes. 4. Convert database(s) to the Rdb 7.2 structure level using RMU/CONVERT. 5. Verify database(s) again using RMU/VERIFY/ALL. 6. Backup database(s) using RMU/BACKUP. 7. Access database(s) from Alpha and Integrity directly by specifying database root file specification(s) in SQL ATTACH statements.
2	Add an Integrity server to an existing mixed cluster of VAX and Alpha nodes and access an Rdb database	1. Verify database(s) using

	from all nodes. Disks used for the database are accessible from all nodes.	RMU/VERIFY/ALL. 2. Backup database(s) using RMU/BACKUP. 3. Install Rdb 7.2 on Integrity and Alpha nodes. 4. Convert database(s) to the Rdb 7.2 structure level using RMU/CONVERT. 5. Verify database(s) again using RMU/VERIFY/ALL. 6. Backup database(s) using RMU/BACKUP. 7. Access database(s) from Alpha and Integrity nodes directly by specifying database root file specification(s) in SQL ATTACH statements. 8. Access the database from VAX node(s) using the Rdb built−in network server (remote database) by specifying one of the Alpha or Integrity node names in SQL ATTACH statements. 9. After thorough testing, remove VAX nodes from the cluster.
3	Move database(s) to new disks and add an Integrity server to an existing cluster.	1. Use RMU/COPY with an options file to move the database files to the new disks. 2. Follow the steps for case 1 or case 2.
4	Continue to use Rdb primarily from VAX or Alpha nodes using earlier releases. Add an Integrity server for application testing purposes.	1. Install Rdb 7.2 on Integrity node. 2. Access existing database(s) from Integrity node by specifying one of the Alpha or VAX node names in the SQL ATTACH statements. 3. When testing is complete, follow the steps in case 1 or case 2.
5	Add an Integrity server to an existing cluster of Alpha servers or Create a new cluster from an existing stand−alone Alpha server by adding one or more new Integrity servers.	1. Verify database(s) using RMU/VERIFY/ALL. 2. Backup database(s) using RMU/BACKUP. 3. Install Rdb 7.2 on Integrity and Alpha nodes.

		4. Convert database(s) to the Rdb 7.2 structure level using RMU/CONVERT. 5. Verify database(s) again using RMU/VERIFY/ALL. 6. Backup database(s) using RMU/BACKUP. 7. Access database(s) from Alpha and Integrity directly by specifying database root file specification in the SQL ATTACH statements.
6	Create a new stand−alone Integrity Server system or cluster of Integrity Servers and move database(s) to the new environment.	1. Verify database(s) using RMU/VERIFY/ALL. 2. Install Rdb 7.2 on new system(s). 3. Back up database(s) on the existing cluster using RMU/BACKUP. 4. Copy backup file(s) to the new system (or, if using tape media, make the tapes available to the new system). 5. Restore database(s) on the new system using RMU/RESTORE specifying the location of each database file in an options file. 6. Verify the new database using RMU/VERIFY/ALL.

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

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

Chapter 2
Software Errors Fixed in Oracle Rdb Release 7.2.4.1

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.4.1.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 Process Termination Due to Register Stack Engine (RSE) Stack Overflow

On Itanium systems, the Register Stack Engine (RSE) stack was not correctly resized when using the logical RDMS$BIND_EXEC_STACK_SIZE to increase Rdb's executive mode stack length. This could result in unexpected process deletions if the RSE stack overflowed. Some large, complex queries (those that, for example, specify a large number of parameters to the IN selection clause) may exceed the default levels of Rdb's executive mode stack. If this occurs, the query will fail with the following messages:

```
%RDB−F−IMP_EXC, facility−specific limit exceeded
−RDMS−F−XPR_STACK_OFLO, expression forces too many levels of recursion
```

This problem has been corrected in Oracle Rdb Release 7.2.4.1. The size of the executive mode stack is adjusted using the logical RDMS$BIND_EXEC_STACK_SIZE. The default size is 500 pagelets. The logical name RDMS$BIND_EXEC_RSE_STACK_SIZE is used to adjust the Register Stack Engine (RSE) stack length. The default size is 1000 pagelets or the value of RDMS$BIND_EXEC_STACK_SIZE, whichever is larger.

2.1.2 Query Using Multiple IN Clauses With DECODE Function Bugchecks

Bug 8651003

The following query, using multiple IN clauses with the DECODE function, bugchecks.

```
SELECT  T1.SEQ,
        T1.AREA,
        T1.REQ
FROM
     T1 T1,
     T2 T2,
     T3 T3
WHERE
    T1.BILL_IND IN
           ((DECODE('V', '', T1.BILL_IND, 'V')),
            (DECODE('V', '', T1.BILL_IND, ' ')))
    AND T2.CORP IN  (' ', ' ')
    AND T1.AREA      = T2.AREA
;
%RDMS−I−BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
%RDMS−I−BUGCHKDMP, generating bugcheck dump file SQLBUGCHK.DMP;
 SQL$721  SQL$SQL  SQL$$FLUSH_INPUT_ON_CONTROL_C
                                         8042 0000000000000804 00000000001511C4
 SQL$721  SQL$SQL  SQL$$FLUSH_INPUT_ON_CONTROL_C
                                         8042 0000000000000804 00000000001511C4
%SYSTEM−F−ACCVIO, access violation, reason mask=00,
virtual address=000000000000000C, PC=000000000028BD5C, PS=0000001B
```

This problem occurs when the query uses multiple IN clauses with the DECODE function in one of the IN clauses.

This problem has been corrected in Oracle Rdb Release 7.2.4.1.

2.1.3 Intermittent RDB−E−NO_DUP Index Field Value Already Exists

Bugs 2892551 and 9032751

With workload collection enabled, it was possible for intermittent "RDB−E−NO_DUP, Index field value already exists; duplicates not allowed for RDB$WRKLD_ID_FLD_NDX" errors to occur.

This problem has been corrected in Oracle Rdb Release 7.2.4.1.

2.1.4 Query Using Bitmap Bugchecks

Bug 9016641

The following query, using bitmap, bugchecks.

```
set flags 'bitmap';
select count(*) from car
    where
        make = 'holden'
        and cyear <> 1978
        and colour = 'red'
        and lplate > 'AAA000'
        and ctype = 'sedan';
%RDMS−I−BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
```

This problem occurs when the query applies 5 predicates using 5 indexes. The query works if any one of the predicates is removed.

This problem has been corrected in Oracle Rdb Release 7.2.4.1.

2.1.5 Wrong Result From Query With Match Strategy

Bug 9124868

The following query with zigzag match strategy returns the wrong result (2 rows instead of 4 rows).

```
SELECT A.ACODE, A.ORD_NO, A.LNK_NO, A.TRAN_NO
    FROM T1_V A, T2 I
  WHERE
         A.ECODE  = 'XETR'
     AND A.TDATE  = DATE'2009−11−10'
     AND A.ECODE  = I.ECODE
     AND A.ACODE  = I.ACODE
     AND TIMESTAMP'2009−11−10 12:00:00.00' BETWEEN I.TS_FROM AND I.TS_TO;
Tables:
  0 = T1
  1 = T2
Conjunct: (0.ECODE = 1.ECODE) AND (0.ACODE = 1.ACODE)
Match  Q1
  Outer loop
  Match_Keys:0.ECODE, 0.ACODE
    Index only retrieval of relation 0:T1
      Index name  T1_NDX [2:2]
        Keys: (0.ECODE = 'XETR') AND (0.TDATE = DATE '2009−11−10')
  Inner loop      (zig−zag)
  Match_Keys:1.ECODE, 1.ACODE
  Index_Keys:ACODE, TS_FROM, TS_TO, ECODE
    Conjunct: TIMESTAMP '2009−11−10 12:00:00.00' >= 1.TS_FROM
    Conjunct: TIMESTAMP '2009−11−10 12:00:00.00' <= 1.TS_TO
    Conjunct: 1.ECODE = 'XETR'
    Index only retrieval of relation 1:T2
      Index name  T2_NDX [0:0]
        Bool: (1.ECODE = 'XETR') AND
              (TIMESTAMP '2009−11−10 12:00:00.00' >= 1.TS_FROM) AND
              (TIMESTAMP '2009−11−10 12:00:00.00' <= 1.TS_TO)
 A.ACODE             A.ORD_NO   A.LNK_NO
   A.TRAN_NO
 GB0007547838      93140000163987520               14739
                        0

 GB0007547838      93140000163987522               14739
                        0

2 rows selected
```

Notice that the match keys are ordered by the trailing index segment key and the leading index segment key of the index T2_NDX, where the leading and trailing segments are separated by some non−matched key(s).

The query works if the index T2_NDX is changed by swapping the leading and trailing segments, as in the following example.

```
drop index T2_NDX;
create unique index T2_NDX
    on T2 (
    ECODE      −− 1st segment
        asc,
    TS_FROM
        desc,
    TS_TO
        asc,
    ACODE
        asc    −− 4th segment
        );

The following are the output traces:

Tables:
  0 = T1
  1 = T2
Conjunct: (0.ECODE = 1.ECODE) AND (0.ACODE = 1.ACODE)
Match  Inner_TTBL Q1
  Outer loop
  Match_Keys:0.ECODE, 0.ACODE
    Index only retrieval of relation 0:T1
      Index name  T1_NDX [2:2]
        Keys: (0.ECODE = 'XETR') AND (0.TDATE = DATE '2009−11−10')
  Inner loop
  Match_Keys:1.ECODE, 1.ACODE
    Temporary relation
    Sort: 1.ECODE(a), 1.ACODE(a)
    Conjunct: TIMESTAMP '2009−11−10 12:00:00.00' <= 1.TS_TO
    Index only retrieval of relation 1:T2
      Index name  T2_NDX [2:1]
        Keys: (1.ECODE = 'XETR') AND
              (TIMESTAMP '2009−11−10 12:00:00.00' >= 1.TS_FROM)
        Bool: TIMESTAMP '2009−11−10 12:00:00.00' <= 1.TS_TO
 A.ACODE              A.ORD_NO   A.LNK_NO
   A.TRAN_NO
 GB0007547838      93140000163987520               14724
                        0

 GB0007547838      93140000163987521               14724
                        0

 GB0007547838      93140000163987520               14739
                        0

 GB0007547838      93140000163987522               14739
                        0
4 rows selected
```

The problem occurs when the inner loop of the match strategy in the query applies the match keys ordered by the trailing index segment key and the leading index segment key of the inner index where the leading and trailing segments are separated by some non−matched key(s). See example below.

```
match keys: D, A
index keys: A, B, C, D
```

The query works if the leading and trailing index segments are swapped to be in the same order as the match keys.

```
match keys: D, A
index keys: D, B, C, A
```

This problem has been corrected in Oracle Rdb Release 7.2.4.1.

2.1.6 Unexpected Error After Truncating a Row Cached Table

Bug 8888176

Under certain conditions, when using row cache, a TRUNCATE TABLE operation might not have been completely successful. The truncate might have appeared to work but subsequent operations might have failed. This occured when the table was stored in a MIXED area with a HASHED index.

After reloading the table, errors may be reported such as:

```
%RDB−E−NO_RECORD, access by dbkey failed because dbkey is no longer associated
with a record
−RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−F−NODBK, 63:2:1 does not point to a data record
```

Or an unexpected query termination could occur.

```
%RDB−E−STREAM_EOF, attempt to fetch past end of record stream
```

This problem has been corrected in Oracle Rdb Release 7.2.4.1.

2.1.7 Wrong Result From Aggregate Match Join With Distinct

Bug 9196565

The following aggregate match join query with DISTINCT clause returns the wrong results (8 rows instead of 15 rows).

```
select * from x c1 where c1.tab in
 (select c2.depends_on from constr_depend c2);
Tables:
  0 = X
  1 = CREL
  2 = CREL
Conjunct: <agg0> <> 0
Match    (Agg Outer Join)  Q1
  Outer loop
  Match_Key:0.TAB
    Get     Retrieval by index of relation 0:X
      Index name  X_RN [0:0]
  Inner loop
  Match_Key:2.RELATION_NAME
    Aggregate: 0:COUNT−ANY (<subselect>) Q2
    Reduce: 1.RELATION_NAME, 2.RELATION_NAME
    Sort: 1.RELATION_NAME(a), 2.RELATION_NAME(a)
    Cross block of 2 entries  Q4
      Cross block entry 1
        Leaf#01 BgrOnly 2:CREL Card=245
          Bool: BITSTRING (2.FLAGS FROM 3 FOR 1) = 1
          BgrNdx1 CREL_RN [0:0] Fan=8
      Cross block entry 2
        Conjunct: (1.CONSTRAINT_NAME = 2.CONSTRAINT_NAME) AND (1.RELATION_NAME
                   <> 2.RELATION_NAME)
        Leaf#02 BgrOnly 1:CREL Card=245
          Bool: BITSTRING (1.FLAGS FROM 3 FOR 1) = 0
          BgrNdx1 CREL_CN [1:1] Fan=8
            Keys: 1.CONSTRAINT_NAME = 2.CONSTRAINT_NAME
 TAB                    DEPENDS_ON
 TC_IC_USEC             T_SEC
 TC_IC_USEC             T_M_CLASS
 TC_IC_USEC             T_CURR
 TC_IC_USEC             TC_IC_SUB
 TC_IC_USEC             TC_IC_EQU
 TC_IC_USEC             TC_IC_BOND
 T_M_CLASS              T_CURR
 T_M_CLASS              TC_IC_USEC
8 rows selected
```

Constr_depend is defined as follows:

```
create view constr_depend (tab, depends_on) as
```

```
select distinct rn0,rn1 from cv ;

create view cv (rn0, rn1) as
select fl0.rn,fl1.rn
    from fl0 join fl1 using (cn)
    where fl0.rn <> fl1.rn ;

create view fl0 (rn, cn) as
select RELATION_NAME,CONSTRAINT_NAME from CREL
where bitstring (flags from 3 for 1) = 0;

create view fl1 (rn, cn) as
select RELATION_NAME,CONSTRAINT_NAME from CREL
where bitstring (flags from 3 for 1) = 1;
```

The query works if the DISTINCT clause is explicitly applied at the main select statement, as in the following example.

```
select distinct * from x c1 where c1.tab in
 (select c2.depends_on from constr_depend c2);
Tables:
  0 = X
  1 = CREL
  2 = CREL
Reduce: 0.TAB, 0.DEPENDS_ON
Sort: 0.TAB(a), 0.DEPENDS_ON(a)
Conjunct: <agg0> <> 0
Match    (Agg Outer Join)  Q1
  Outer loop
  Match_Key:0.TAB
    Get     Retrieval by index of relation 0:X
      Index name  X_RN [0:0]
  Inner loop
  Match_Key:2.RELATION_NAME
    Aggregate: 0:COUNT-ANY (<subselect>) Q2
    Cross block of 2 entries  Q4
      Cross block entry 1
        Conjunct: BITSTRING (2.FLAGS FROM 3 FOR 1) = 1
        Get     Retrieval by index of relation 2:CREL
          Index name  CREL_RN [0:0]
      Cross block entry 2
        Conjunct: (1.CONSTRAINT_NAME = 2.CONSTRAINT_NAME) AND (1.RELATION_NAME
                   <> 2.RELATION_NAME)
        Leaf#01 BgrOnly 1:CREL Card=245
          Bool: BITSTRING (1.FLAGS FROM 3 FOR 1) = 0
          BgrNdx1 CREL_CN [1:1] Fan=8
            Keys: 1.CONSTRAINT_NAME = 2.CONSTRAINT_NAME
 TAB                   DEPENDS_ON
 TC_IC_BND             TC_IC_USEC
 TC_IC_EQU             TC_IC_USEC
 TC_IC_MEM             TC_IC_CLPRC
 TC_IC_MEM             TC_IC_MEM_MAP
 TC_IC_MEM_MAP         TC_IC_MEM
 TC_IC_MEM_MAP         TC_IC_TLOC
 TC_IC_SUB             TC_IC_USEC
 TC_IC_USEC            TC_IC_BND
 TC_IC_USEC            TC_IC_EQU
 TC_IC_USEC            TC_IC_SUB
 TC_IC_USEC            T_CURR
 TC_IC_USEC            T_M_CLASS
 TC_IC_USEC            T_SEC
 T_M_CLASS             TC_IC_USEC
 T_M_CLASS             T_CURR
15 rows selected
```

The problem occurs when the query with distinct clause joins by match strategy with the join key being a subset of the order of the inner stream.

In the following strategy output, the match key "2.RELATION_NAME" is a subset of the order of the DISTINCT clause "1.RELATION_NAME(a), 2.RELATION_NAME(a)".

```
Conjunct: <agg0> <> 0
Match    (Agg Outer Join)  Q1
  Outer loop
  Match_Key:0.TAB
    Get     Retrieval by index of relation 0:X
      Index name  X_RN [0:0]
  Inner loop
  Match_Key:2.RELATION_NAME                   <== this is the match key
    Aggregate: 0:COUNT-ANY (<subselect>) Q2
    Reduce: 1.RELATION_NAME, 2.RELATION_NAME     <== Reduce for DISTINCT
    Sort: 1.RELATION_NAME(a), 2.RELATION_NAME(a) <== this is sort order
    Cross block of 2 entries  Q4
      Cross block entry 1
        Leaf#01 BgrOnly 2:CREL Card=245
          Bool: BITSTRING (2.FLAGS FROM 3 FOR 1) = 1
          BgrNdx1 CREL_RN [0:0] Fan=8
      Cross block entry 2
        Conjunct: (1.CONSTRAINT_NAME = 2.CONSTRAINT_NAME) AND (1.RELATION_NAME
                   <> 2.RELATION_NAME)
        Leaf#02 BgrOnly 1:CREL Card=245
          Bool: BITSTRING (1.FLAGS FROM 3 FOR 1) = 0
          BgrNdx1 CREL_CN [1:1] Fan=8
            Keys: 1.CONSTRAINT_NAME = 2.CONSTRAINT_NAME
```

This problem has been corrected in Oracle Rdb Release 7.2.4.1.

2.1.8 Using RDB Remote Created a Logfile Called RDBSERVER.EXE

Bugs 5600820 and 2544477

When using Rdb Remote with TCP/IP from one node to another and with RDBSERVER defined in the LNM$SYSTEM_TABLE and with the account RDB$REMOTEnn or RDB$REMOTE (depending on if you use MULTIVERSION or STANDARD) having enough privileges to write in SYS$SYSTEM, then a new version of the file RDBSERVERnn.EXE or RDBSERVER.EXE was created on the remote node. However, the new version of the file did not have the contents of the original RDBSERVER(nn).EXE but contained the contents of the RDBSERVER.LOG file.

This problem has been corrected in Oracle Rdb Release 7.2.4.1. The RDBSERVER TCP/IP Service now creates the log file RDBSERVER_TCPIP.LOG.

IMPORTANT NOTE:

In a cluster environment, the service must be DISABLED and then ENABLED on all cluster members other than the one used for the installation for this update to be complete.

2.1.9 Wrong Result From LIMIT TO Query With ORDER BY DESC

Bug 9361560

The following LIMIT TO query with ORDER BY DESC returns the wrong result (1 row instead of 0 rows).

```
SELECT C0.YMD, C0.KBN
  FROM CALENDER C0
  WHERE
    C0.YMD = '20100319' and
    NOT EXISTS(SELECT 1
               FROM CALENDER C1,
                 (SELECT C2.YMD, C2.KBN
                   FROM CALENDER2 C2
                   WHERE
                      C2.YMD <= C1.YMD
                   ORDER BY C2.YMD DESC
                   LIMIT TO 1 ROWS) Q3
               WHERE Q3.YMD = C0.YMD)
;
Tables:
  0 = CALENDER
  1 = CALENDER
  2 = CALENDER2
Cross block of 2 entries  Q1
  Cross block entry 1
    Index only retrieval of relation 0:CALENDER
      Index name  IDX_CALENDER_1 [1:1]
        Keys: 0.CALENDER_YMD = '20100319'
  Cross block entry 2
    Conjunct: <agg0> = 0
    Aggregate−F1: 0:COUNT−ANY (<subselect>) Q2
    Cross block of 2 entries  Q2
      Cross block entry 1
        Index only retrieval of relation 1:CALENDER
          Index name  IDX_CALENDER_0 [0:0]
      Cross block entry 2
        Conjunct: 2.YMD = 0.YMD
        Merge of 1 entries  Q2
          Merge block entry 1  Q3
          Firstn: 1
          Index only retrieval of relation 2:CALENDER2
            Index name  IDX_CALENDER2_1 [0:1]
              Keys: 2.YMD <= 1.YMD
 YMD            KBN
 20100319       1
1 row selected
```

The query works in the following cases:

1. TRANSITIVITY is disabled
2. LIMIT TO clause is removed

Two possible workarounds are shown in the following example.

```
Workaround 1:
------------
set flags 'NOTRANSITIVITY'
<--- execute the above query --->
Tables:
  0 = CALENDER
  1 = CALENDER
  2 = CALENDER2
Cross block of 2 entries  Q1
  Cross block entry 1
    Index only retrieval of relation 0:CALENDER
      Index name  IDX_CALENDER_1 [1:1]
        Keys: 0.CALENDER_YMD = '20100319'
  Cross block entry 2
    Conjunct: <agg0> = 0
    Aggregate-F1: 0:COUNT-ANY (<subselect>) Q2
    Cross block of 2 entries  Q2
      Cross block entry 1
        Index only retrieval of relation 1:CALENDER
          Index name  IDX_CALENDER_0 [0:0]
      Cross block entry 2
        Conjunct: 2.YMD = 0.YMD
        Merge of 1 entries  Q2
          Merge block entry 1  Q3
          Firstn: 1
          Index only retrieval of relation 2:CALENDER2
            Index name  IDX_CALENDER2_1 [0:1]      Reverse Scan
              Keys: 2.YMD <= 1.YMD
0 rows selected

Workaround 2:
------------
! LIMIT TO is removed
!
SELECT C0.YMD, C0.KBN
  FROM CALENDER C0
  WHERE
    C0.YMD = '20100319' and
    NOT EXISTS(SELECT 1
               FROM CALENDER C1,
                 (SELECT C2.YMD, C2.KBN
                   FROM CALENDER2 C2
                   WHERE
                      C2.YMD <= C1.YMD
                   ORDER BY C2.YMD DESC
                   -- LIMIT TO 1 ROWS       <== removed
                   ) Q3
               WHERE Q3.YMD = C0.YMD)
;
Tables:
  0 = CALENDER
  1 = CALENDER
  2 = CALENDER2
Cross block of 2 entries  Q1
  Cross block entry 1
    Index only retrieval of relation 0:CALENDER
      Index name  IDX_CALENDER_1 [1:1]
        Keys: 0.CALENDER_YMD = '20100319'
  Cross block entry 2
    Conjunct: <agg0> = 0
    Aggregate-F1: 0:COUNT-ANY (<subselect>) Q2
    Cross block of 2 entries  Q2
```

```
      Cross block entry 1
        Index only retrieval of relation 1:CALENDER
          Index name  IDX_CALENDER_0 [0:0]
      Cross block entry 2
        Conjunct: 2.YMD = 0.YMD
        Merge of 1 entries  Q2
          Merge block entry 1  Q3
          Conjunct: 2.YMD <= 1.YMD
          Index only retrieval of relation 2:CALENDER2
            Index name  IDX_CALENDER2_1 [1:1]
              Keys: 2.YMD = 0.YMD
0 rows selected
```

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

1. The main query contains an aggregate (e.g. NOT EXIST) and a filter predicate where the column of one of the filter predicates is used later in a transitive equality predicate.
2. The aggregate subquery joins another subselect query using the transitive equality predicate.
3. The subselect query contains ORDER BY <column of transitive equality> DESC clause followed by LIMIT TO.

This problem has been corrected in Oracle Rdb Release 7.2.4.1.

2.1.10 Wrong Result From OR Predicate With Constants as Index Boolean

Bug 9509316

A query returns the wrong result (0 rows instead of 3 rows) when the operand of the OR predicate is mapped to the constant column of the derived table.

```
Here is the list of rows in the tables:

select SUB_TYP_COD, TYP_COD from T2 order by 1;
 SUB_TYP_COD   TYP_COD
 CCF           BON
 CPF           BON
2 rows selected

select SUB_TYP_COD, TYP_COD from T1 order by 1;
 SUB_TYP_COD   TYP_COD
 CCF           BON
 CCF           BON
 CPF           BON
3 rows selected

SET FLAGS 'MAX_STAB';   ! disable dynamic optimizer
select V2.INST_ID, C_CONST
from
   (select
    T1.INST_ID,
    T1.FRT_DAT,
    T1.FIN_TYP,
    '001'
    from
        T2, T1
    where
```

```
        (T2.TYP_COD = T1.TYP_COD) AND
        (T2.SUB_TYP_COD = T1.SUB_TYP_COD) AND
        (T1.LST_DAT >= DATE '2002−10−04')
        ) as V2 (INST_ID, FRT_DAT, FIN_TYP, C_CONST)
where
    (v2.c_const = '001' or v2.c_const = '001')
    and (v2.c_const = '001' or v2.c_const = '001')
    ;
Tables:
  0 = T2
  1 = T1
Conjunct: (('001' = '001') OR ('001' = '001')) AND (('001' = '001') OR ('001' =
          '001')) AND ('001' = '001')
Merge of 1 entries  Q1
  Merge block entry 1  Q2
  Cross block of 2 entries  Q2
    Cross block entry 1
      Get     Retrieval by index of relation 1:T1
        Index name  T1_NDX [1:0]
          Keys: 1.LST_DAT >= DATE '2002−10−04'
          Bool: (('001' = '001') OR ('001' = '001')) AND (('001' = '001') OR (
                '001' = '001')) AND ('001' = '001')
    Cross block entry 2
      Conjunct: 0.TYP_COD = 1.TYP_COD
      Get     Retrieval by index of relation 0:T2
        Index name  T2_NDX [1:1]
          Keys: 0.SUB_TYP_COD = 1.SUB_TYP_COD
          Bool: (('001' = '001') OR ('001' = '001')) AND (('001' = '001') OR (
                '001' = '001')) AND ('001' = '001')
0 rows selected
```

The query returns the correct results (3 rows) if the OR predicates are replaced by AND's, as in the following example.

```
SET FLAGS 'NOMAX_STAB';     ! enable dynamic optimizer
select V2.INST_ID, C_CONST
from
   (select
    T1.INST_ID,
    T1.FRT_DAT,
    T1.FIN_TYP,
    '001'
    from
        T2, T1
    where
        (T2.TYP_COD = T1.TYP_COD) AND
        (T2.SUB_TYP_COD = T1.SUB_TYP_COD) AND
        (T1.LST_DAT >= DATE '2002−10−04')
        ) as V2 (INST_ID, FRT_DAT, FIN_TYP, C_CONST)
where
    (v2.c_const = '001' AND v2.c_const = '001')
    and (v2.c_const = '001' AND v2.c_const = '001')
    ;
Tables:
  0 = T2
  1 = T1
Merge of 1 entries  Q1
  Merge block entry 1  Q2
  Cross block of 2 entries  Q2
    Cross block entry 1
      Get     Retrieval by index of relation 1:T1
```

```
        Index name  T1_NDX [1:0]
          Keys: 1.LST_DAT >= DATE '2002-10-04'
          Bool: ('001' = '001') AND ('001' = '001') AND ('001' = '001') AND (
                '001' = '001')
    Cross block entry 2
      Conjunct: 0.TYP_COD = 1.TYP_COD
      Get     Retrieval by index of relation 0:T2
        Index name  T2_NDX [1:1]
          Keys: 0.SUB_TYP_COD = 1.SUB_TYP_COD
          Bool: ('001' = '001') AND ('001' = '001') AND ('001' = '001') AND (
                '001' = '001')
 INST_ID    C_CONST
 C00002     001
 100905     001
 110036     001
3 rows selected
```

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

1. The main query selects from the derived table where one of the columns is a constant.
2. The main WHERE clause contains an OR predicate where the left hand side operand is mapped to the constant column of the derived table.
3. The OR predicates are generated as index boolean "Bool:".

This problem has been corrected in Oracle Rdb Release 7.2.4.1.

2.2 SQL Errors Fixed

2.2.1 Not all Errors Were Written to Log File Created by SET OUTPUT

Bug 8911782

In prior versions of Oracle Rdb, the error reported when an indirect command file could not be opened was not written to the log file created by the SET OUTPUT command. This is shown by the following example.

```
SQL>set out DISK:[USER.LOG]create_view.log
SQL>@DISK:[USER.LOG]CREATE_VIEW.LOG
%COSI−F−FILACCERR, error opening file DISK:[USER.SQL]V_TABLE.SQL;
−RMS−E−FNF, file not found

$ search DISK:[USER.LOG]CREATE_VIEW.LOG "%"
%SEARCH−I−NOMATCHES, no strings matched
```

This means that applications which process the output created by SET OUTPUT may not detect a failure as expected. This problem has been corrected in Oracle Rdb Release 7.2.4.1.

2.2.2 Cardinality Changes Lost by ALTER STORAGE MAP

Bug 5689817

In prior releases of Oracle Rdb, the update of the cardinality counters for a table would be lost when an ALTER STORAGE MAP statement was executed prior to the COMMENT and after rows were deletd or inserted.

The following example shows a significant change to the table cardinality which results from the TRUNCATE TABLE statement. Normally, such a command would leave the table with RDB$CARDINALITY equal to zero. However, the update to this column is deferred until COMMIT time and is discarded by the ALTER STORAGE MAP statement.

```
SQL> set trans read write reserving SALARY_HISTORY
cont> for exclusive write;
SQL>
SQL> truncate table SALARY_HISTORY;
SQL> alter storage map SALARY_HISTORY_MAP store in jobs;
SQL>
SQL> commit;
SQL>
SQL> select count(*) from SALARY_HISTORY;

           0
1 row selected
SQL> select rdb$cardinality
cont>  from rdb$relations
cont>  where rdb$relation_name = 'SALARY_HISTORY';
      RDB$CARDINALITY
                  729
1 row selected
```

```
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.4.1. Rdb now correctly preserves the cardinality changes during the ALTER STORAGE MAP statement.

2.2.3 CHECK Constraint for Declared Variables Not Supported on Integrity Systems

Bug 8984248

The CHECK clause for variables declared in a compound statement were not supported on Integrity systems. Usage of such syntax resulted in a bugcheck dump.

The following example shows a simple example and the resulting bugcheck dump.

```
SQL> create module MOD_CRASH
cont>     language SQL
cont>
cont>     procedure P_CRASH_DB (in :I CHAR(1));
cont>     begin atomic
cont>     declare :J CHAR(1) = :I
cont>        check(value in ('X','O','')) not deferrable;
cont>     trace :j, :i;
cont>     end;
cont> end module;
SQL> commit;
SQL>
SQL> call P_CRASH_DB('A');
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]RDSBUGCHK.DMP;
```

This problem has been corrected in Oracle Rdb Release 7.2.4.1. The implementation of this feature is now complete on Integrity systems.

2.2.4 Unexpected Bugcheck When Dropping a LIST OF BYTE VARYING Column

Bug 8994347

In prior releases of Oracle Rdb, it was possible that ALTER TABLE ... DROP COLUMN would generate a bugcheck if the following conditions were true:

- The column was of type LIST OF BYTE VARYING.
- The table appeared in a LIST storage map definition.
- No columns were explicitly referenced by the LIST storage map for this table.

The following example shows results using the MF_PERSONNEL database. The LIST storage map for this database is defined as follows.

```
create storage map LISTS_MAP
    store LISTS
        in RESUME_LISTS
        for (
                RESUMES)
            in RDB$SYSTEM;
```

ALTER TABLE generates the error.

```
$ SQL$
SQL> attach 'filename MF_PERSONNEL';
SQL> alter table resumes drop column resume;
%RDMS−I−BUGCHKDMP, generating bugcheck dump file USER1:[TESTING]RDSBUGCHK.DMP;
```

This problem has been corrected in Oracle Rdb Release 7.2.4.1.

2.2.5 Unexpected SQL−E−TRUN−STORE Error When Using UPPER or LOWER Function

Bug 9007038

In prior releases of Oracle Rdb, it was possible that SQL would under−size the markers passed to functions in a Dynamic SQL statement. The affected functions were UPPER, LOWER, SUBSTRING, TRANSLATE, LEAST and GREATEST. This problem involved the processing of VARCHAR types for these functions and could lead to unexpected errors such as:

```
%SQL−E−TRUN_STORE, String truncated during assignment to a column
```

This problem has been corrected in Oracle Rdb Release 7.2.4.1. SQL now computes better estimates for these functions when the source type is not known.

2.2.6 Unexpected INVALID_BLR Error From ALTER TABLE ... ALTER COLUMN

Bug 9299822

In prior versions of Oracle Rdb, it was possible for an ALTER TABLE ... ALTER COLUMN to fail as shown in the following example.

```
SQL> set flags 'VIEW_RECOMPILE';
SQL>
SQL> create view TEST_VIEW (sal_amt) as
cont>     select distinct
cont>         (select abs (salary_amount−1)
cont>          from salary_history
cont>          where employee_id = jh.employee_id and jh.job_end is null)
cont>     from job_history jh
cont>     where employee_id = '00000';
SQL>
SQL> alter table salary_history
cont>     alter column SALARY_AMOUNT bigint(3);
~MI: View name "TEST_VIEW", New Version #2
%RDB−E−NO_META_UPDATE, metadata update failed
−RDB−E−INVALID_BLR, request BLR is incorrect at offset 72
SQL>
```

This would occur when a view was dependent on the table and column being altered, and that view column was defined as part of a SELECT DISTINCT and included a conditional expression such as CASE, NULLIF, COALESCE, NVL, NVL2, ABS or SIGN. When Oracle Rdb attempted to recompile the view definition to use the revised data type, it did not correctly handle these conditional expressions within a DISTINCT clause.

This problem has been corrected in Oracle Rdb Release 7.2.4.1.

2.2.7 Unexpected Bugcheck Reporting RDMS−E−SIGNATURE_MISMA, Invalid Parameter Signature on Procedure Call

Bug 9206054

It was possible in some cases that an ATTACH or CONNECT to a database could fail with the error RDMS−E−SIGNATURE_MISMA, invalid parameter signature on procedure call.

This problem has been corrected in Oracle Rdb Release 7.2.4.1.

2.3 RDO and RDML Errors Fixed

2.3.1 RDO IMPORT Alignment Faults

Previously, the RDO IMPORT command could generate a significant number of alignment faults while processing the input file.

This problem has been reduced in Oracle Rdb Release 7.2.4.1. The RDO IMPORT command correctly accesses aligned data buffers which serves to reduce alignment faults.

2.4 RMU Errors Fixed

2.4.1 RMU/RECOVER/ORDER_AIJ_FILES Fails With RMS−E−FNF When Processing One After Image Journal File

If there is only one after image journal file and you try to recover the database using the /Order_Aij_Files qualifier, it fails to find the AIJ file.

If there is only one AIJ file, then there is no need to prune the list with the /Order_Aij_Files qualifier.

This problem has been corrected in Oracle Rdb Release 7.2.4.1.

2.5 Row Cache Errors Fixed

2.5.1 RMU /POPULATE_CACHE Not Correctly Fetching System Records

Bug 9019973

Previously, when using the RMU /POPULATE_CACHE /INDEX= command to insert hashed index nodes into a row cache, system records in the mixed−format storage area were not correctly added to the cache. In particuar, an incorrect logical database key was used to access the system records causing them to be inserted into the cache with the incorrect logical database keys. This, in turn, would cause the system records to not be located when they were later accessed, giving the appearance that they had not been inserted into the cache resulting in unexpected database page IO.

This issue has been corrected in Oracle Rdb Release 7.2.4.1. RMU now correctly readies both the hashed index and system record logical areas and accesses the system records with the correctly formed logical database key.

System Records Inserted Only For Pages With Index Nodes

Note that when using the RMU /POPULATE_CACHE /INDEX= command to insert hashed index nodes into a row cache, system records will only be inserted into the cache for those database pages that contain hashed index nodes. The system record will not be inserted into cache if a page has no hashed index nodes.

Chapter 3
Software Errors Fixed in Oracle Rdb Release 7.2.4.0

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.4.

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 Bugcheck With SYSTEM−F−ROPRAND in KODTXN$POST_TSNBLK_UPDATE

Bugs 7193991 and 8796832

It is possible for a process to bugcheck due to a corrupt internal queue header with a reserved operand fault in the routine KODTXN$POST_TSNBLK_UPDATE. Correcting the memory corruption requires closing and re−opening the database. This problem will generally present itself with a bugcheck exception "footprint" similar to the following:

```
***** Exception at 00000000815C89D2 : RDMSHRP72\KODTXN$POST_TSNBLK_UPDATE
+ 000000D2
%SYSTEM−F−ROPRAND, reserved operand fault at PC=00000000815C89D2, PS=00000009
Saved PC = 00000000815C3EF0 : RDMSHRP72\KOD$START + 00000D40
```

Analysis of the bugcheck dump file will often indicate that at least one of the "TUPB_RELQHD" queue headers will contain an entry containing "00000000:00000001" as in the following example where entry 13 has been corrupted:

```
TUPB_RELQHD_VEC[1.] @04910A00 = 00000000:00000000 (04910A00:04910A00)
TUPB_RELQHD_VEC[2.] @04910A08 = 00000000:00000000 (04910A08:04910A08)
TUPB_RELQHD_VEC[3.] @04910A10 = 00000000:00000000 (04910A10:04910A10)
TUPB_RELQHD_VEC[4.] @04910A18 = 00000000:00000000 (04910A18:04910A18)
TUPB_RELQHD_VEC[5.] @04910A20 = 00000000:00000000 (04910A20:04910A20)
TUPB_RELQHD_VEC[6.] @04910A28 = 00000000:00000000 (04910A28:04910A28)
TUPB_RELQHD_VEC[7.] @04910A30 = 00000000:00000000 (04910A30:04910A30)
TUPB_RELQHD_VEC[8.] @04910A38 = 00000000:00000000 (04910A38:04910A38)
TUPB_RELQHD_VEC[9.] @04910A40 = 00000000:00000000 (04910A40:04910A40)
TUPB_RELQHD_VEC[10.] @04910A48 = 00000000:00000000 (04910A48:04910A48)
TUPB_RELQHD_VEC[11.] @04910A50 = 00000000:00000000 (04910A50:04910A50)
TUPB_RELQHD_VEC[12.] @04910A58 = 00000000:00000000 (04910A58:04910A58)
TUPB_RELQHD_VEC[13.] @04910A60 = 00000000:00000001 (04910A60:04910A61)
```

The cause of the problem was related to an incorrect synchronization between processes manipulating a relative memory queue within the database global section. This (or related) problem may impact all Rdb databases that perform transactions by more than one database user. The possibility of corruption increases with higher transaction rates and when there are more database users performing transactions. Generally, databases configured for less than 168 users will never see this symptom and databases with fewer than 168 simultaneous users will never see this problem.

Oracle recommends that all Rdb installations upgrade to at least Oracle Rdb Release 7.2.4 to implement the correction to this problem.

This problem has been corrected in Oracle Rdb Release 7.2.4. The shared memory access to "TUPB_RELQHD" queue headers is now correctly synchronized.

3.1.2 Memory Leak On Systems With RAD Support Enabled

Bug 8410893

When running on an Alpha system with RAD support enabled, a memory leak was possible during database detach/attach sequences. The size of the leak would be related to the size of the per−RAD database statistics global section.

As a possible workaround, the system parameter RAD_SUPPORT can be set to zero.

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.1.3 Incorrect Messages From RMU /MONITOR START When RDM$MON_USERNAME Specifies Non−existant Account

Bug 8420114

In prior releases, when the logical name RDM$MON_USERNAME was defined to specify a non−existant username, the RMU /MONITOR START command would return incorrect or misleading messages. The following example shows such output:

```
$ RMU /MONITOR START
%RMU−F−CANTCREMON, unable to start database monitor process
%F
%?−RESULTOVF
%F−CTRLERR
%I−POWERFAIL
−PLI
%NOMSG, Message number 0053474E
%BADPARAM
%NORMAL
Message number 00081C48
−INPSMB
−JBC, normal successful completion
−TRACE, Message number 0009A930
%F−NOMSG, Message number 4F525024
```

As a workaround, either do not define RDM$MON_USERNAME or make sure that it specifies a valid username.

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.1.4 Potential INVEXCEPTN System Crash Using LARGE MEMORY IS ENABLED or SHARED MEMORY IS SYSTEM on Itanium

Bug 8541571

When closing a database when using the SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED feature on some Itanium systems, it was possible for Oracle Rdb to cause an OpenVMS system crash with a bugcheck type of "INVEXCEPTN, Exception while above ASTDEL". The INVEXCEPTN can be triggered by an invalid access to the OpenVMS PTE database.

The following OpenVMS crash "footprint" is a result of this problem:

```
Bugcheck Type:     INVEXCEPTN, Exception while above ASTDEL
Node:              RDBI64   (Cluster)
CPU Type:          HP rx8640  (1.60GHz/12.0MB)
VMS Version:       V8.3−1H1
Current Process:   RDMS_MONITOR72
Current Image:     $1$DGA4000:[SYS2.SYSCOMMON.][SYSEXE]RDMMON72.EXE;3
Failing PC:        00000000.0057DDD0    DEALLOC_PHY_BUFFER_C+001C0
Failing PS:        00000000.00000803
Module:            RDMPRV72
Offset:            000A7DD0
```

This problem only effects those Itanium systems with physical memory accessible to the system of PFNs greater than 268,435,455 (Hex 0FFFFFFF). The SDA command CLUE CONFIG can be used to display the system memory configuration.

The workaround to this problem for such systems with PFNs greater than 268,435,455, is to discontinue use of the LARGE MEMORY IS ENABLED and SHARED MEMORY IS SYSTEM features, if enabled, by using one, or both, of the following SQL commands:

- ALTER DATABASE ... GLOBAL BUFFERS ...LARGE MEMORY IS DISABLED
- ALTER DATABASE ... SHARED MEMORY IS PROCESS RESIDENT

This problem has been corrected in Oracle Rdb Release 7.2.4. The Oracle Rdb VLM feature correctly processes 64−bit PFN values when closing databases using the SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED feature on Itanium systems.

3.1.5 Wrong Result From Query With Derived Table

Bug 8475230

The following query returns the wrong result: 4 rows instead of 5 rows.

```
SELECT TRANS_DATE
FROM
(SELECT TRANS_DATE
 FROM     T1 T1
    WHERE
        T1.TRANS_DATE    <= '20081231' AND
        T1.TRANS_DATE    >  '20081222' AND
        T1.TRANS_DATE NOT IN
        (
         SELECT TRANS_DATE
            FROM T2
            WHERE
                T2_FLAG = 'Y' AND
                CODE = 'FBAR' AND
                TRANS_DATE >  '20081222' AND
                TRANS_DATE <= '20081231'
        )
        AND
        T1.TRANS_DATE NOT IN
        (
         SELECT TRANS_DATE
            FROM V1             -- another view V1
            WHERE
                IS_FLAG ='Y' AND
                CODE ='FBAR' AND
                TRANS_DATE > '20081222' AND
                TRANS_DATE <='20081231'
        )
 ORDER BY T1.TRANS_DATE DESC) DT;
Tables:
  0 = T1
  1 = T2
  2 = T2
  3 = T3
  4 = T4
  5 = T3
Merge of 1 entries  Q1
  Merge block entry 1  Q2
  Cross block of 3 entries  Q2
    Cross block entry 1
      Get     Retrieval by index of relation 0:T1
        Index name  T1_IDX [1:1]       Reverse Scan
          Keys: (0.TRANS_DATE > '20081222') AND (0.TRANS_DATE <=
                '20081231')
    Cross block entry 2
      Conjunct: <agg0> = 0
      Aggregate-F1: 0:COUNT-ANY (<subselect>) Q3
      Conjunct: (1.T2_FLAG = 'Y') AND (1.CODE = 'FBAR') AND (MISSING (
                0.TRANS_DATE) OR MISSING (1.TRANS_DATE) OR (
                0.TRANS_DATE = 1.TRANS_DATE))
      Get     Retrieval by index of relation 1:T2
        Index name  T2_IDX [1:1]
          Keys: (1.TRANS_DATE > '20081222') AND (1.TRANS_DATE <=
                '20081231')
    Cross block entry 3
      Conjunct: <agg1> = 0
      Aggregate-F1: 1:COUNT-ANY (<subselect>) Q4
      Conjunct: (CASE (WHEN (Q6-CAGG"=AGG_TOTAL"<var0> = Q6-CAGG"=AGG_TOTAL"
                <var1>) THEN 'Y' ELSE 'N') = 'Y') AND (2.CODE = 'FBAR')
                 AND (2.TRANS_DATE > '20081222') AND (2.TRANS_DATE <=
                '20081231') AND (MISSING (0.TRANS_DATE) OR MISSING (
                2.TRANS_DATE) OR (0.TRANS_DATE = 2.TRANS_DATE))
      Aggregate: 2:SUM (<var1>) Q6
                 3:SUM (<var0>) Q6
      Cross block of 3 entries  Q6
        Cross block entry 1
          Get     Retrieval by index of relation 2:T2   <--See NOTE
            Index name  T2_IDX [1:1]
              Keys: (2.TRANS_DATE > '20081222') AND (2.TRANS_DATE <=
                    '20081231')
        Cross block entry 2
          Merge of 1 entries  Q6
            Merge block entry 1  Q9
            Aggregate: 4:COUNT (*) Q10
            Conjunct: (5.DONE = 'Y') AND (5.STATUS = 'N')
            Get     Retrieval by index of relation 5:T3
              Index name  T3_IDX [3:3]
                Keys: (2.TRANS_DATE = 5.TRANS_DATE) AND (2.CITY = 5.CITY) AND
```

```
                    (2.TOWN = 5.TOWN)
       Cross block entry 3
         Merge of 1 entries  Q6
           Merge block entry 1  Q7
           Aggregate: 5:COUNT (*) Q8
           Cross block of 2 entries  Q8
             Cross block entry 1
               Conjunct: 4.NON_FLAG = 'N'
               Conjunct: 4.NON_REASON = 'P'
               Get     Retrieval by index of relation 4:T4
                 Index name  T4_IDX [1:1]
                   Keys: 2.TRANS_DATE = 4.CURR_DATE
             Cross block entry 2
               Conjunct: (2.MARKET = 3.MARKET) AND (2.CITY = 3.CITY) AND
                         (2.TOWN = 3.TOWN) AND (3.DONE = 'Y') AND
                         (3.STATUS = 'N')
               Get     Retrieval by index of relation 3:T3
                 Index name  T3_IDX [2:2]    Direct lookup
                   Keys: (2.TRANS_DATE = 3.TRANS_DATE) AND (3.PRODUCT =
                         4.PRODUCT)
 TRANS_DATE
 20081230
 20081229
 20081224
 20081223
4 rows selected
```

NOTE:: The conjuncts are missing to filter the retrieval of context "2:T2" under Cross block entry 1 and causes the query to return the wrong result.

The query works if the derived table is removed from the query, as in the following example.

```
SELECT TRANS_DATE
 FROM     T1 T1
    WHERE
        T1.TRANS_DATE    <= '20081231' AND
        T1.TRANS_DATE    >  '20081222' AND
        T1.TRANS_DATE NOT IN
        (
         SELECT TRANS_DATE
            FROM T2
            WHERE
                T2_FLAG = 'Y' AND
                CODE = 'FBAR' AND
                TRANS_DATE >  '20081222' AND
                TRANS_DATE <= '20081231'
        )
        AND
        T1.TRANS_DATE NOT IN
        (
         SELECT TRANS_DATE
            FROM V1             -- another view V1
            WHERE
                IS_FLAG ='Y' AND
                CODE ='FBAR' AND
                TRANS_DATE > '20081222' AND
                TRANS_DATE <='20081231'
        )
 ORDER BY T1.TRANS_DATE DESC;
Tables:
  0 = T1
  1 = T2
  2 = T2
  3 = T3
  4 = T4
  5 = T3
Cross block of 3 entries  Q1
  Cross block entry 1
    Get     Retrieval by index of relation 0:T1
      Index name  T1_IDX [1:1]       Reverse Scan
        Keys: (0.TRANS_DATE > '20081222') AND (0.TRANS_DATE <= '20081231')
  Cross block entry 2
    Conjunct: <agg0> = 0
    Aggregate−F1: 0:COUNT−ANY (<subselect>) Q2
    Conjunct: (1.T2_FLAG = 'Y') AND (1.CODE = 'FBAR') AND (MISSING (
              0.TRANS_DATE) OR MISSING (1.TRANS_DATE) OR (0.TRANS_DATE
               = 1.TRANS_DATE))
    Get     Retrieval by index of relation 1:T2
      Index name  T2_IDX [1:1]
        Keys: (1.TRANS_DATE > '20081222') AND (1.TRANS_DATE <= '20081231')
  Cross block entry 3
    Conjunct: <agg1> = 0
    Aggregate−F1: 1:COUNT−ANY (<subselect>) Q3
    Conjunct: (CASE (WHEN (Q5−CAGG"=AGG_TOTAL"<var0> = Q5−CAGG"=AGG_TOTAL"<var1>
              ) THEN 'Y' ELSE 'N') = 'Y') AND (MISSING (0.TRANS_DATE) OR
              MISSING (2.TRANS_DATE) OR (0.TRANS_DATE = 2.TRANS_DATE))
    Aggregate: 2:SUM (<var1>) Q5
               3:SUM (<var0>) Q5
    Cross block of 3 entries  Q5
      Cross block entry 1
        Conjunct: (2.CODE = 'FBAR') AND (2.TRANS_DATE > '20081222')
                   AND (2.TRANS_DATE <= '20081231')     <−− See NOTE
        Get     Retrieval by index of relation 2:T2
          Index name  T2_IDX [1:1]
            Keys: (2.TRANS_DATE > '20081222') AND (2.TRANS_DATE <=
                  '20081231')
      Cross block entry 2
        Merge of 1 entries  Q5
          Merge block entry 1  Q8
          Aggregate: 4:COUNT (*) Q9
          Conjunct: (5.DONE = 'Y') AND (5.STATUS = 'N')
          Get     Retrieval by index of relation 5:T3
            Index name  T3_IDX [3:3]
              Keys: (2.TRANS_DATE = 5.TRANS_DATE) AND (2.CITY = 5.CITY) AND
                    (2.TOWN = 5.TOWN)
      Cross block entry 3
        Merge of 1 entries  Q5
          Merge block entry 1  Q6
          Aggregate: 5:COUNT (*) Q7
          Cross block of 2 entries  Q7
            Cross block entry 1
              Conjunct: 4.NON_FLAG = 'N'
              Conjunct: 4.NON_REASON = 'P'
              Get     Retrieval by index of relation 4:T4
                Index name  T4_IDX [1:1]
                  Keys: 2.TRANS_DATE = 4.CURR_DATE
            Cross block entry 2
              Conjunct: (2.MARKET = 3.MARKET) AND (2.CITY = 3.CITY) AND
                        (2.TOWN = 3.TOWN) AND (3.DONE = 'Y') AND
                        (3.STATUS = 'N')
              Get     Retrieval by index of relation 3:T3
                Index name  T3_IDX [2:2]    Direct lookup
                  Keys: (2.TRANS_DATE = 3.TRANS_DATE) AND (3.PRODUCT =
                       4.PRODUCT)
 TRANS_DATE
 20081231
 20081230
 20081229
 20081224
 20081223
5 rows selected
```

NOTE:: The conjuncts appear correctly in the retrieval of context "2:T2" under Cross block entry 1.

The workaround is to replace the "NOT IN" clause with "NOT EXISTS" in the query, as in the following example.

```
SELECT TRANS_DATE
FROM
(SELECT TRANS_DATE
 FROM     T1 T1
    WHERE
        T1.TRANS_DATE    <= '20081231' AND
        T1.TRANS_DATE    >  '20081222' AND
        T1.TRANS_DATE NOT IN
        (
         SELECT TRANS_DATE
            FROM T2
            WHERE
                T2_FLAG = 'Y' AND
                CODE = 'FBAR' AND
                TRANS_DATE >  '20081222' AND
                TRANS_DATE <= '20081231'
        )
        AND
        T1.TRANS_DATE NOT IN
        (
         SELECT TRANS_DATE
            FROM V1             -- another view V1
            WHERE
                IS_FLAG ='Y' AND
                CODE ='FBAR' AND
                TRANS_DATE > '20081222' AND
                TRANS_DATE <='20081231'
        )
 ORDER BY T1.TRANS_DATE DESC) DT;
Tables:
  0 = T1
  1 = T2
  2 = T2
  3 = T3
  4 = T4
  5 = T3
Cross block of 3 entries  Q1
  Cross block entry 1
    Aggregate-F1: 0:COUNT-ANY (<subselect>) Q4
    Conjunct: CASE (WHEN (Q6-CAGG"=AGG_TOTAL"<var0> = Q6-CAGG"=AGG_TOTAL"<var1>)
               THEN 'Y' ELSE 'N') = 'Y'
    Aggregate: 1:SUM (<var1>) Q6
               2:SUM (<var0>) Q6
    Cross block of 3 entries  Q6
      Cross block entry 1
        Conjunct: (2.CODE = 'FBAR') AND (2.TRANS_DATE > '20081222')
                   AND (2.TRANS_DATE <= '20081231')    <-- See NOTE
        Get     Retrieval by index of relation 2:T2
          Index name  T2_IDX [1:1]
            Keys: (2.TRANS_DATE > '20081222') AND (2.TRANS_DATE <=
                  '20081231')
      Cross block entry 2
        Merge of 1 entries  Q6
          Merge block entry 1  Q9
          Aggregate: 3:COUNT (*) Q10
          Conjunct: (5.T2_FLAG = 'Y') AND (5.STATUS = 'N')
          Get     Retrieval by index of relation 5:T3
            Index name  T3_IDX [3:3]
              Keys: (2.TRANS_DATE = 5.TRANS_DATE) AND (2.CITY =
                     5.CITY) AND (2.TOWN = 5.TOWN)
      Cross block entry 3
        Merge of 1 entries  Q6
          Merge block entry 1  Q7
          Aggregate: 4:COUNT (*) Q8
          Cross block of 2 entries  Q8
            Cross block entry 1
              Conjunct: 4.NON_FLAG = 'N'
              Conjunct: 4.NON_REASON = 'P'
              Get     Retrieval by index of relation 4:T4
                Index name  T4_IDX [1:1]
                  Keys: 2.TRANS_DATE = 4.CCUR_DATE
            Cross block entry 2
              Conjunct: (2.MARKET = 3.MARKET) AND (2.CITY =
                        3.CITY) AND (2.TOWN = 3.TOWN
                        ) AND (3.T2_FLAG = 'Y') AND (3.STATUS = 'N')
              Get     Retrieval by index of relation 3:T3
                Index name  T3_IDX [2:2]    Direct lookup
                  Keys: (2.TRANS_DATE = 3.TRANS_DATE) AND (3.PRODUCT =
                        4.PRODUCT)
  Cross block entry 2
    Aggregate−F1: 5:COUNT−ANY (<subselect>) Q3
    Conjunct: (1.HOLIDAY = 'Y') AND (1.CODE = 'FBAR')    <−− See NOTE
    Get     Retrieval by index of relation 1:T2
      Index name  T2_IDX [1:1]
        Keys: (1.TRANS_DATE > '20081222') AND (1.TRANS_DATE <= '20081231')
  Cross block entry 3
    Merge of 1 entries  Q1
      Merge block entry 1  Q2
      Conjunct: (<agg5> = 0) AND (<agg0> = 0)
      Get     Retrieval by index of relation 0:T1
        Index name  T1_IDX [1:1]       Reverse Scan
          Keys: (0.TRANS_DATE > '20081222') AND (0.TRANS_DATE <=
                '20081231')
 TRANS_DATE
 20081231
 20081230
 20081229
 20081224
 20081223
5 rows selected
```

NOTE:: The conjuncts appear correctly in the retrieval of context "2:T2" and "1:T2".

This problem was introduced in Oracle Rdb Release 7.2.3.0.

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.1.6 Application Looses Virtual Memory (Memory Leak)

Bugs 7697133 and 8449605

An application could run out of virtual memory or pagefile quota when executing multiple queries within a single attach to a database and when these queries (or stored procedures) accessed indices.

This happened only on Integrity systems and was caused by allocating some small index key structures from a memory pool which is only freed at the end of the session (during database disconnect).

This loss of virtual memory increases if SAMPLED SELECTIVITY is used (OPTIMIZE clause of a query, SET FLAGS 'SELECTIVITY(2), or the SET OPTIMIZATION LEVEL statement).

These problems have been corrected in Oracle Rdb Release 7.2.4.

3.1.7 VMS$BUFFER_OBJECT_USER Not Always Checked

Bug 8464087

In some cases, the OpenVMS rights identifier VMS$BUFFER_OBJECT_USER was not being required for database buffer object use. This could allow users to utilize the buffer objects feature even though they did not have the identifier granted.

This problem has been corrected in Oracle Rdb Release 7.2.4. Users attempting to utilize buffer object features now must always have the VMS$BUFFER_OBJECT_USER rights identifier granted.

It is possible that applications that worked previously when using the buffer objects feature with processes that did not have the VMS$BUFFER_OBJECT_USER rights identifier granted may now correctly fail with messages similar to the following:

```
%SQL-F-ERRATTDEF, Could not use database file specified by SQL$DATABASE
-RDB-F-SYS_REQUEST, error from system services request
-RDMS-F-CANTCREBOB, error creating Buffer Object
-SYSTEM-E-NOBUFOBJID, requires rights identifier VMS$BUFFER_OBJECT_USER
```

These messages indicate that the process must be granted the VMS$BUFFER_OBJECT_USER rights identifier before attempting to use the buffer objects feature. In many cases, granting the identifier to the user account and then logging in again will resolve the issue.

3.1.8 Deleted Space in Uniform Areas Not Reclaimed by Other Users When Erased Rows Moved From Row Cache to Disk

Bug 8522094

Oracle Rdb Release 7.2 introduced a mechanism that allows database users on the same cluster node to share information regarding the availability of free space. When a user chooses a location to store new rows, the location is stored in the database global section so that other users can use that location as a starting point when searching for available space. When a user deletes rows from a table, if the location of the deleted rows is closer to the beginning of the storage area than the last page used for an insert, then the starting page for the next insert is updated to the location of the lowest page that had rows deleted.

In some cases when using the Row Cache feature, when an erased row was moved from cache back to a page in the database, the shared information regarding the availability of free space was not being updated. This caused users to be unaware of possible free space and unexpected storage area extention could result.

This problem has been corrected in Oracle Rdb Release 7.2.4. The shared information regarding the availability of free space is now updated when erased rows are moved from a row cache back to the database page.

3.1.9 LOWER Function Problem When Using ISOLATINCYRILLIC Character Set

Bug 8605022

A problem in the casing tables for ISOLATINCYRILLIC would produce wrong lowercase values for the following two cyrillic characters:

- UPPERCASE ER (hex C0)
- UPPERCASE SHA (hex C8)

This problem has been corrected in Oracle Rdb Release 7.2.4. The LOWER function now produces the correct lowercase characters for ISOLATINCYRILLIC.

3.1.10 Query Bugchecks with SYSTEM−F−FLTINV

Bugs 8580585 and 7649113

The following query bugchecks with SYSTEM−F−FLTINV error when sampled selectivity is applied.

```
  SELECT DISTINCT 7
   FROM  R_ILS_STATUS S , R_LS L
   WHERE S.A_DATE>=20090531  AND S.A_DATE<=20090531
   AND  S.A_NLS = L.A_NLS
   AND NOT EXISTS
        ( SELECT 1 FROM R_ILS I where I.A_NLS=S.a_nls
                    AND I.A_DOP=s.a_date )
   OPTIMIZE WITH SAMPLED SELECTIVITY    ;
%RDB−E−ARITH_EXCEPT, truncation of a numeric value at runtime
−SYSTEM−F−HPARITH, high performance arithmetic trap, Imask=00000000,
Fmask=00000001, summary=02, PC=000000000008D0FB8, PS=0000001B
−SYSTEM−F−FLTINV, floating invalid operation, PC=000000000008D0FB8, PS=0000001B
```

The query works if sampled selectivity is NOT applied, as in the following example.

```
  SELECT DISTINCT 7
   FROM  R_ILS_STATUS S , R_LS L
   WHERE S.A_DATE>=20090531  AND S.A_DATE<=20090531
   AND  S.A_NLS = L.A_NLS
   AND NOT EXISTS
        ( SELECT 1 FROM R_ILS I where I.A_NLS=S.a_nls
                    AND I.A_DOP=s.a_date )
   ! OPTIMIZE WITH SAMPLED SELECTIVITY
   ;
Tables:
  0 = R_ILS_STATUS
  1 = R_LS
  2 = R_ILS
Reduce: 7
Sort: 7(a)
Cross block of 3 entries  Q1
  Cross block entry 1
    Index only retrieval of relation 1:R_LS
      Index name  SI_LS_9 [0:0]
  Cross block entry 2
    Leaf#01 BgrOnly 0:R_ILS_STATUS Card=608708
      Bool: (0.A_DATE >= 20090531) AND (0.A_DATE <= 20090531) AND (0.A_NLS =
            1.A_NLS)
      BgrNdx1 I_R_ILS_STATUS [1:1] Fan=13
        Keys: 0.A_NLS = 1.A_NLS
      BgrNdx2 I_R_ILS_STATUS2 [1:1] Fan=17
        Keys: (0.A_DATE >= 20090531) AND (0.A_DATE <= 20090531)
  Cross block entry 3
    Conjunct: <agg0> = 0
    Aggregate−F1: 0:COUNT−ANY (<subselect>) Q2
    Index only retrieval of relation 2:R_ILS
      Index name  SI_ILS_NDK [2:2]   Index counts lookup
        Keys: (2.A_NLS = 0.A_NLS) AND (2.A_DOP = 0.A_DATE)
0 rows selected
```

This problem occurs when the database contains empty tables or indices and the query uses the optimizer with sampled selectivity.

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.1.11 Bugcheck Altering Storage Map

Bug 8573340

A bugcheck could occur when executing an alter storage map using thresholds and compression.

This problem only occurred under OpenVMS Itanium.

A typical alter storage map SQL statement:

```
        ALTER STORAGE MAP STORE_MAP
            STORE IN DATA001_AREA
                (THRESHOLDS ARE (80,90,95))
            ENABLE COMPRESSION
            REORGANIZE;

***** Exception at 000000008095F620 : RDMSHRP721\RDMS$$EXE_NEXT + 00002710
%COSI−F−BUGCHECK, internal consistency failure
```

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.1.12 Wrong Results when Expression Shared Between Aggregate Filters

Bug 8553416

In prior versions of Oracle Rdb V7.2, queries that included shared expressions between aggregate filters might return the wrong results.

The following example shows such a query with the expression "substring(postal_code from 1 for 3)" shared between the two count filters.

```
select
    count(*) filter
        (where
            substring(postal_code from 1 for 1) <> '0'
            or substring(postal_code from 3 for 1) <> '3'
            or substring(postal_code from 1 for 3) = '034'
         ) as field_a
    ,count(*) filter
        (where
            employee_id < '00300' and
            substring(postal_code from 1 for 3) <> '034'
        ) as field_b
from
    employees e, departments d
where
    e.employee_id = d.manager_id
    and d.department_code starting with 'SU';
```

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.1.13 Optimize For Sequential Access Using Index

Bug 8746157

In prior releases of Oracle Rdb, some UPDATE statements would ignore the "Optimize for Sequential Access" clause. See the following example.

```
UPDATE T1 SET VALUE = 'A'
FROM T1
        WHERE   ID       = 1
        AND     COUNTRY  = 12
        AND     MARKET   = 20
        AND     GROUP    = 6
        AND     TRAN_NO  = 5710175
OPTIMIZE FOR SEQUENTIAL ACCESS;
Get     Temporary relation      Retrieval by index of relation 0:T1
  Index name  T1_IDX [5:5]    Direct lookup
~S: Full compliance with the outline was not possible
```

This problem has been corrected in Oracle Rdb Release 7.2.4. Rdb now correctly processes the "Optimize for Sequential Access" in such cases.

3.1.14 Query With Match Strategy for Left Outer Join Bugchecks

Bug 8709430

In prior releases of Oracle Rdb, it was possible for queries with match strategy for a left outer join operation to generate a bugcheck dump with the following footprint.

```
***** Exception at 000000000688627C : RDMSHRP72\RDMS$$EXE_CREATE_TTBL_FILE +
0000214C
%SYSTEM−F−ACCVIO, access violation, reason mask=00, virtual address=..., PC=
...., PS=...
Saved PC = 0000000006880CEC : RDMSHRP72\RDMS$$EXE_NEXT + 0000110C
Saved PC = 000000000687FFD0 : RDMSHRP72\RDMS$$EXE_NEXT + 000003F0
Saved PC = 000000000688445C : RDMSHRP72\RDMS$$EXE_CREATE_TTBL_FILE + 0000032C
Saved PC = 000000000688451C : RDMSHRP72\RDMS$$EXE_CREATE_TTBL_FILE + 000003EC
Saved PC = 00000000068800D8 : RDMSHRP72\RDMS$$EXE_NEXT + 000004F8
Saved PC = 0000000006880808 : RDMSHRP72\RDMS$$EXE_NEXT + 00000C28
```

An example query with these similar characteristics follows.

```
select *
from
  (select
    (select count(*)
     from T1 t1 left outer join T2 t2
     on (t2.COL1 = t1.COL1) and (t2.COL2 = t1.COL2)
     where (t1.COL3 = c2.COL3) and (t1.COL4 = 0)
     )
     ,c2.*
    ,0
  from T5 c2
  )
  −−as c1 (f0, f2,  f3)
where
    (1 = 1) and (COL5 = 1);
%RDMS−I−BUGCHKDMP, generating bugcheck dump file [directory]RDSBUGCHK.DMP;
```

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.1.15 Bugchecks at DIOFETCH$FETCH_SNAP_SEG or DIOCCH$FETCH_SNAP_SEG

In some cases, when using read−only transactions along with read−write transactions that request exclusive access to a table, the read−only transaction may bugcheck with a "footprint" similar to the following:

```
***** Exception at 0000000001948194 : RDMSHRP72\DIOFETCH$FETCH_SNAP_SEG
+ 000003C4
%COSI−F−BUGCHECK, internal consistency failure
Saved PC = 00000000019488B4 : RDMSHRP72\DIOFETCH$FETCH_VISIBLE_SEG + 00000684
Saved PC = 00000000019492FC : RDMSHRP72\DIOFETCH$FETCH_ONE_LINE + 000008AC
Saved PC = 0000000001949CCC : RDMSHRP72\DIOFETCH$SCAN_ONE_PAGE + 000002AC
Saved PC = 000000000194A228 : RDMSHRP72\DIO$FETCH_AREA + 00000228
Saved PC = 000000000174616C : RDMSHRP72\RDMS$$EXE_NEXT + 00001D9C
Saved PC = 0000000001744A24 : RDMSHRP72\RDMS$$EXE_NEXT + 00000654
Saved PC = 0000000001746748 : RDMSHRP72\RDMS$$EXE_NEXT + 00002378
Saved PC = 0000000002081F04 : symbol not found
Saved PC = 000000000175F9C8 : RDMSHRP72\RDMS$TOP_START_REQUEST + 00000878
```

This problem would only occur on databases with a transaction sequence number larger than 2,147,483,647. The following shows an example sequence that could lead to the bugcheck:

```
 SESSION 1:                          SESSION 2:

      SQL$
      SQL> ATTACH 'FILE FOO';
      SQL> SET TRANS READ ONLY;
      SQL> SHOW TRANSACTION;

                                     SQL$
                                     SQL> ATTACH 'FILE FOO';
                                     SQL> SET TRANS READ WRITE
                                             RESER C1 FOR EXCL WRITE;
                                     SQL> SHOW TRANSACTION;
                                     SQL> DELETE FROM C1;
                                     SQL> COMMIT;

      SQL> SELECT * FROM C1;
      %RDMS−I−BUGCHKDMP
```

This problem has been corrected in Oracle Rdb Release 7.2.4. The invalid access by the read−only transaction is correctly detected and the expected error is generated:

```
SQL> SELECT * FROM C1;
%RDB−F−LOCK_CONFLICT, request failed due to locked resource
−RDMS−F−CANTSNAP, can't ready storage area $1$DGA1:[DB]FOO.RDB;1 for snapshots
SQL>
```

3.1.16 Bugcheck in PSII2INSERTDUPBBC

In prior versions of Oracle Rdb, a problem in establishing correct index node currency during the insertion of a duplicate record dbkey into a sorted ranked index might result in a bugcheck with a "footprint" similar to the following:

```
***** Exception at 0000000000321770 : RDMSHRP72\PSII2INSERTDUPBBC +00002560
%COSI−F−BUGCHECK, internal consistency failure
Saved PC = 000000000031A950 : SQLU721\PSII2INSERTBOTTOM + 00000B80
Saved PC = 0000000000301980 : SQLU721\PSII2INSERTT + 00000400
Saved PC = 0000000000304770 : SQLU721\PSII2INSERTTREE + 00000450
```

As a possible workaround, consider utilizing a sorted index rather than a sorted ranked index.

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.2 SQL Errors Fixed

3.2.1 Unexpected Failures of TRUNC and ROUND Functions

Bug 6945515

In prior versions of Oracle Rdb, the ROUND and TRUNC functions could fail with an UNSNUMXPR (Unsupported numeric expression) error. The following example shows a query generated by an Oracle RDBMS tool. Here the :1 is a parameter to the dynamic query.

```
SELECT "D_CUST_CODE","D_ORD_NO","D_READY_DATE" FROM "DOR_ORDER" WHERE
"D_READY_DATE">=ADD_MONTHS(LAST_DAY(TRUNC(:1))+1,−1)
ERROR at line 6:
ORA−32800: internal error [No corresponding Oracle message for Rdb error]
%SQL−F−UNSNUMXPR, Unsupported numeric expression
```

The error from Oracle Rdb indicates that it defaulted to a numeric TRUNC function.

This problem has been corrected in Oracle Rdb Release 7.2.4. SQL now does a better job of predicting the data type of input parameters based on secondary arguments to ROUND and TRUNC and also from the function context.

3.2.2 Unexpected Restrictions in Date/Time Subtraction

Bug 2946256

In prior versions of Oracle Rdb, the use of TIMESTAMP and DATE VMS in subtraction expressions was restricted and resulted in either DATESUBILL or DATESCANEQ errors. The following examples show these problems.

- DATE VMS could not be used in a subtraction expression without being CAST as a TIMESTAMP. In this example, DVMS is defined as DATE VMS, and TS is defined as TIMESTAMP.

  ```
  select
      (dvms − ts) year(7) to month
  from
      dt_table;
  %SQL−F−UNSDATXPR, Unsupported date expression
  −SQL−F−DATESUBILL, Operands of date/time subtraction are incorrect
  ```
- TIMESTAMP types with different fractional second precision could not appear in a subtraction expression. In this example, TS1 is defined as TIMESTAMP(1).

  ```
  select
      (ts − ts1) day(7) to second(2)
  from
      dt_table;
  %SQL−F−UNSDATXPR, Unsupported date expression
  −SQL−F−DATESCANEQ, Date/time expressions with different
  fractional seconds precision are not comparable
  ```
- Expressions that should have resulted in a number of days (ORACLE LEVEL1 or ORACLE2 dialects only) were not permitted. In this example, BIRTHDAY is defined as TIMESTAMP(0) which is not

compatible with SYSDATE which is type DATE VMS.

```
set dialect 'oracle level2';
select (sysdate-bday)*(60*24) from foo;
%SQL-F-UNSDATXPR, Unsupported date expression
-SQL-F-DATESUBILL, Operands of date/time subtraction are incorrect
```

These restrictions have been lifted in Oracle Rdb Release 7.2.4. For the purposes of subtraction, DATE VMS is treated as a TIMESTAMP(2) and differences in fractional second precision are no longer considered incompatibilities between these types.

3.2.3 Unexpected DEFVALINC Error When Altering a Column's Domain

Bug 8242912

In prior versions of Oracle Rdb, the ALTER TABLE ... ALTER COLUMN command could fail when altering a domain for a column. This occurred because the old DEFAULT for the column was compared to the older (and in this case smaller) DEFAULT instead of the new DEFAULT inherited from the new domain.

The following example shows the unexpected error.

```
SQL> create domain char_12 char(12) default 'short str';
SQL> create domain char_20 char(20) default 'this is longer';
SQL> create table mytab (col1 char_20);
SQL> alter table mytab alter column col1 char_12;
%SQL-W-CHR_TOO_SHO, Character length of column COL1 is too short
%SQL-F-DEFVALINC, You specified a default value for COL1 which
is inconsistent with its data type
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.4. In this release, no error is reported although a warning will be issued because of the reduced size of the column.

3.2.4 Unexpected FLDNOTCRS Error When Referencing DBKEY or ROWID in Quotes

Bug 8294794

In prior versions of Oracle Rdb, a column reference to DBKEY or ROWID within quotes would result in a FLDNOTCRS error. The following example shows the error.

```
SQL> select a."ROWID" from rdb$database a;
%SQL-F-FLDNOTCRS, Column A.ROWID was not found in the tables in current scope
SQL>
```

When either SET DIALECT or SET QUOTING RULES enable quoting of columns, the references to the pseudo columns DBKEY or ROWID should be permitted.

This problem has been corrected in Oracle Rdb Release 7.2.4. SQL now allows the pseudo columns DBKEY or ROWID to be quoted. They must be in all upper case letters.

3.2.5 Unexpected Bugcheck When Assigning Value to FOR Cursor Variables

Bug 8429420

In prior versions of Oracle Rdb, attempts to assign a value to a FOR cursor variable would cause a bugcheck. These variables are read−only copies of the select expressions and may not be used as targets for SELECT ... INTO clause, UPDATE ... RETURNING clause and the INSERT ... RETURNING clause.

```
SQL> create module M
cont>
cont> procedure P;
cont> begin
cont> for :emp as each row of
cont>     select employee_id, last_name from employees
cont> do
cont>     select last_name
cont>     into :emp.last_name
cont>     from employees
cont>     where employee_id = :emp.employee_id;
cont> end for;
cont> end;
cont>
cont> end module;
%RDMS−I−BUGCHKDMP, generating bugcheck dump file DISK1:[TEST]SQLBUGCHK.DMP;
 SQL$721  SQL$SQL  SQL$$FLUSH_INPUT_ON_CONTROL_C
                                          8000 0000000000000798 0000000000150358
 SQL$721  SQL$SQL  SQL$$FLUSH_INPUT_ON_CONTROL_C
                                          8000 0000000000000798 0000000000150358
%SQL−F−BUGCHK, There has been a fatal error. Please contact your Oracle support
representative. SQL$SEMMSG − MERGE_PROC_MSG found
parsym not in signature
```

A similar bugcheck is reported for a multistatement procedure.

```
SQL> begin
cont> for :emp as each row of
cont>     select employee_id, last_name from employees
cont> do
cont>     select last_name
cont>     into :emp.last_name
cont>     from employees
cont>     where employee_id = :emp.employee_id;
cont> end for;
cont> end;
%RDMS−I−BUGCHKDMP, generating bugcheck dump file DISK1:[TEST]SQLBUGCHK.DMP;
 SQL$721  SQL$SQL  SQL$$FLUSH_INPUT_ON_CONTROL_C
                                          8016 00000000000007C4 0000000000150384
 SQL$721  SQL$SQL  SQL$$FLUSH_INPUT_ON_CONTROL_C
                                          8016 00000000000007C4 0000000000150384
%SQL−F−BUGCHK, There has been a fatal error. Please contact your Oracle support
representative. SQL$$BLR_MSG_FIELD_REF − NF and HV in two messages
```

This problem has been corrected in Oracle Rdb Release 7.2.4. SQL now correctly diagnoses such illegal usage, as shown in the following example.

```
SQL> begin
```

```
cont> for :emp as each row of
cont>     select employee_id, last_name from employees
cont> do
cont>     select last_name
cont>     into :emp.last_name
cont>     from employees
cont>     where employee_id = :emp.employee_id;
cont> end for;
cont> end;
%SQL−F−FORFCHVARRO, FOR statement variable "LAST_NAME" is read only
```

3.2.6 Unexpected Bugcheck When Executing External Routine

Bug 8231715

In prior versions of Oracle Rdb V7.2 on OpenVMS Integrity systems, a bugcheck might occur when a stored procedure or stored function attempted to execute an external routine to which the user did not have EXECUTE access.

The following example shows a stored function (CALL_FN) in the module CALLED_MODULE that executes a call to the external procedure LIB$SIGNAL. This external procedure denies access to the current user and should result in the error: %RDB−E−NO_PRIV, privilege denied by database facility.

```
SQL> show privilege on procedure LIB$SIGNAL;
Privileges on Procedure LIB$SIGNAL
    (IDENTIFIER=[RDB,TESTER],ACCESS=NONE)
SQL> show privilege on module CALLED_MODULE;
Privileges on Module CALLED_MODULE
    (IDENTIFIER=[RDB,TESTER],ACCESS=EXECUTE+SHOW+ALTER+DROP+REFERENCES)
SQL>
SQL> set flags 'trace';
SQL> begin
cont> declare :a integer;
cont> set :a = CALL_FN (100);
cont> end;
%RDMS−I−BUGCHKDMP, generating bugcheck dump file
DISK1:[TESTER]RDSBUGCHK.DMP_MINI
%RDMS−I−BUGCHKDMP, generating bugcheck dump file
DISK1:[TESTER]RDSBUGCHK.DMP_MINI
%RDMS−I−BUGCHKDMP, generating bugcheck dump file DISK1:[TESTER]RDSBUGCHK.DMP;
```

This problem has been corrected in Oracle Rdb Release 7.2.4. Oracle Rdb now correctly performs the authorization check while running the stored routine.

3.2.7 Unexpected Error When Importing a Database With Profile Definitions

In prior versions of Oracle Rdb, the SQL EXPORT DATABASE and IMPORT DATABASE commands did not handle correctly databases that contained objects created using the CREATE PROFILE. These objects were incorrectly imported as roles (which would fail) and assignments of profiles to users was not preserved.

The following example shows the errors that are reported by IMPORT.

```
SQL> import database
cont>     from abc_ex
cont>     filename a_ex
cont>     trace;
IMPORTing Users and Roles
Completed CDD$SYSTEM. DIO = 19, CPU = 0:00:00.01, FAULTS = 7
Completed SQLNET4RDB. DIO = 6, CPU = 0:00:00.00, FAULTS = 0
Completed RDB_EXECUTE. DIO = 14, CPU = 0:00:00.01, FAULTS = 6
Completed RDBUSER2. DIO = 13, CPU = 0:00:00.00, FAULTS = 6
%SQL−F−NOPRFRES, unable to import profile DEVELOPMENT_USER
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−E−SECNOTINT, database security checking is not internal
Completed DEVELOPMENT_USER. DIO = 6, CPU = 0:00:00.00, FAULTS = 10
%SQL−F−NOPRFRES, unable to import profile QUERY_USER
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−E−SECNOTINT, database security checking is not internal
Completed QUERY_USER. DIO = 7, CPU = 0:00:00.00, FAULTS = 7
IMPORTing Granted Users and Roles
%SQL−F−NOPGPRES, unable to import granted profile for grantee RDB_EXECUTE
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−E−SECNOTINT, database security checking is not internal
Completed RDB_EXECUTE. DIO = 5, CPU = 0:00:00.00, FAULTS = 3
%SQL−F−NOPGPRES, unable to import granted profile for grantee RDBUSER2
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−E−SECNOTINT, database security checking is not internal
Completed RDBUSER2. DIO = 5, CPU = 0:00:00.00, FAULTS = 0
.
.
.
Completed import. DIO = 1287, CPU = 0:00:00.18, FAULTS = 790
```

This problem has been corrected in Oracle Rdb Release 7.2.4. SQL EXPORT now correctly preserves the assigned profile for each user and IMPORT correctly rebuilds the assigned role list for each user.

3.2.8 Unexpected Bugcheck When Table Partitions Used in RESERVING Clause

Bug 5861614

In prior releases of Oracle Rdb, a bugcheck could result if two tables had partitions that reside in the same storage area. This occurred if a partition of one table was reserved for EXCLUSIVE WRITE and a partition of the other table was reserved for PROTECTED WRITE or SHARED WRITE, as in the following example.

```
SQL> set transaction
cont>     read write
cont>     reserving
cont>        EMPLOYEES partition (1) for EXCLUSIVE WRITE,
cont>        JOB_HISTORY partition (1) for PROTECTED WRITE;
SQL>
SQL> insert into EMPLOYEES
cont>     (employee_id, last_name, first_name, middle_initial)
cont>     value ('00100', 'Jones', 'Fred', 'E');
1 row inserted
SQL>
SQL> insert into JOB_HISTORY
cont>     (employee_id)
cont>     select employee_id from EMPLOYEES where employee_id = '00100';
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTER]RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTER]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=
0000000000000010, PC=000000000028944C, PS=0000001B
```

When using the PARTITION clause and reserving that partition for EXCLUSIVE WRITE, the underlying storage area is reserved for EXCLUSIVE WRITE also and therefore all other partitions that reference this storage area must also be promoted to EXCLUSIVE WRITE.

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.2.9 Unexpected Failure of GRANT and REVOKE When Using Synonyms

Bug 8485872

In prior versions of Oracle Rdb, none of the commands GRANT, REVOKE, SHOW PROTECTION and SHOW PRIVILEGES supported the use of synonyms. The synonyms can be created explicitly by the CREATE SYNONYM command or implicitly by the RENAME command.

The following example shows the problems that were reported:

```
SQL> create table t1 ( a1 int );
SQL> create synonym s1 for t1;
SQL> grant update on s1 to public;
%RDB-E-NO_RECORD, access by dbkey failed because dbkey is no longer associated
with a record
-RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-NODBK, 0:314:7824 does not point to a data record
SQL> show protection on t1;
Protection on Table T1
    (IDENTIFIER=[RDB,TESTER],ACCESS=SELECT+INSERT+UPDATE+DELETE+SHOW+CREATE+
      ALTER+DROP+DBCTRL+REFERENCES)
    (IDENTIFIER=[*,*],ACCESS=NONE)
SQL> show protection on s1;
Protection on Table S1
%SQL-F-RELNOTDEF, Table S1 is not defined in database or schema
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.2.10 Unexpected Correlation Names Appearing as Column Headers

Bug 8648202

In prior releases of Oracle Rdb V7.2, an interactive SQL select which includes a join and also concatenates a zero length string with a column would include just the correlation name in the column header. In prior releases, this would be left blank.

The following example shows the problem.

```
SQL> select ''||a.first_name, ''||b.last_name
cont> from employees a, employees b
cont> limit to 1 row;
 A.           B.
 Terry        Ames
1 row selected
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.4. SQL now suppresses the column header as in prior versions.

3.2.11 Wrong Results When CONCAT Used in an Aggregate Expression with DISTINCT Clause

Bug 8644211

In prior versions of Oracle Rdb V7.2, queries that included CONCAT (or ||) in an aggregate expression with a DISTINCT clause might return the wrong results.

The following example shows such a query.

```
SELECT T.COL4,
       COUNT (DISTINCT ET.COLB4 || ET.COLB6) AS C2
FROM MYTABLE T LEFT OUTER JOIN MYTABLE_DET ET
    ON ET.COLB1 = T.COL1 AND ET.COLB3 = 'E'
WHERE T.COL6 BETWEEN '0' AND '5'
GROUP BY T.COL4;
```

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.2.12 UPDATE ONLY CURSOR Clause Quietly Ignored by SQL Module Language Processor and SQL Precompiler

Bug 7758451

In prior versions of Oracle Rdb, the SQL Module Language processor and SQL precompiler would quietly ignore the UPDATE ONLY CURSOR clause for a DECLARE cursor if it did not detect any update actions for that cursor. That is, no UPDATE ... WHERE CURRENT OF or DELETE ... WHERE CURRENT OF clauses referenced that cursor name.

However, in some cases, the UPDATE ONLY CURSOR clause is used to encourage stricter locking on the table during processing. In many cases, the number of lock operations can be reduced by first locking for update in preparation for the subsequent update, rather than locking first for READ and later upgrading the lock to an UPDATE lock on the row.

This problem has been corrected in Oracle Rdb Release 7.2.4. When such modules are recompiled with this release of Oracle Rdb, the clause UPDATE ONLY CURSOR will no longer be ignored and will be used by Oracle Rdb.

3.3 RMU Errors Fixed

3.3.1 Application Hangs Using Automatic AIJ Backups

Using automatic AIJ backups, the application hangs waiting for the next AIJ to become available. A dump of the database shows inaccessible AIJ files.

```
$ RMU/DUMP db
...
      − 1 journal is inaccessible
          AIJ backup not possible
...
          File is inaccessible
            journal has been made inaccessible by system
            journal is not empty
...
```

The database dump may also show other symptoms of failed or stalled AIJ backups.

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.3.2 Failed RMU MOVE or COPY Deletes Wrong Files

Bug 8400242

A failed RMU MOVE or COPY operation could delete the wrong version of a created database file at the target location. For example, this can happen if, during the RMU operation, a target device becomes full.

For this to happen. there have to be other files in the target location with the exact same filenames as the current database.

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.3.3 RMU Extract Now Extracts Comments for Constraints

Bug 8238128

In prior versions of Oracle Rdb, the RMU Extract command did not extract the comments applied to a constraint by the COMMENT ON CONSTRAINT statement. The following example shows that an additional COMMENT ON statement is output after the CREATE TABLE statement.

```
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename ABC';
create table TESTCON (
    T1K
        SMALLINT
        constraint T1K_PRIMARY
```

```
            primary key
            initially deferred deferrable
        comment is
          'Primary',
    T2
        SMALLINT);

comment on constraint T1K_PRIMARY is
  'Primary key';

commit work;
```

This problem has been corrected in Oracle Rdb Release 7.2.4. RMU Extract now extracts the constraint comments.

3.3.4 Bugcheck in RMU /COLLECT OPTIMIZER_STATISTICS /STATISTICS=WORKLOAD

Bug 8433938

The following command bugchecks:

```
   $ RMU/COLLECT OPTIMIZER_STATISTICS /STATISTICS=WORKLOAD databasename
    %SYSTEM−F−ACCVIO, access violation, reason mask=00, virtual
    address=0000000000000000, PC=000000008002CA10, PS=0000001B
    %RMU−F−FATALOSI, Fatal error from the Operating System Interface.
    %RMU−I−BUGCHKDMP, generating bugcheck dump file DISK:[DIR]RMUBUGCHK.DMP;
    %RMU−F−FTL_COL_STAT, Fatal error for COLLECT OPTIMIZER_STATISTICS
    operation at 14−APR−2009 17:59:08.77
```

This problem is caused by the code accessing data beyond the last entry of a collision error table.

The workaround would be to delete the offending workload column groups.

```
$ RMU/Delete Optimizer_Statistics/Column=(COMPANY_ID, LEGAL_ENTITY_ID, −
GROUP_ID)/Table= CRIT_GROUP database
$ RMU/Delete Optimizer_Statistics/Column=(COMPANY_ID, LEGAL_ENTITY_ID, −
CPTY_MEDIA_IND, GROUP_ID)/Table= CRIT_GROUP database
$ RMU/Delete Optimizer_Statistics/Column=(COMPANY_ID, LEGAL_ENTITY_ID, −
CPTY_MEDIA_IND, GROUP_ID, RECORD_STATE)/Table= CRIT_GROUP database
```

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.3.5 Increased Cardinality Limit in RMU /COLLECT OPTIMIZER_STATISTICS /STATISTICS=WORKLOAD

Bug 8460493

In prior versions of Oracle Rdb, Optimizer Workload Collection in RMU ignored any table with cardinality higher than 249.95 million. See the following example.

```
$RMU/COLLECT OPTIMIZER_STATISTICS /STATISTICS=WORKLOAD TESTDB/LOG
Start loading tables... at 23−APR−2009 09:55:49.41
Done loading tables.... at 23−APR−2009 09:55:49.42
```

```
Start collecting workload stats... at 23−APR−2009 09:55:53.47
Maximum memory required (bytes) = 1166796
Table : XTEST exceeds 249.95 million rows; default statistics values used.
Done collecting workload stats.... at 23−APR−2009 09:55:53.51
Start calculating stats... at 23−APR−2009 09:55:53.51
Done calculating stats.... at 23−APR−2009 09:55:53.51
Start writing stats... at 23−APR−2009 09:55:55.38

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

Optimizer Statistics collected for table : FOO
  Workload Column group  : COL1
  Duplicity factor       : 1.0000000
  Null factor            : 0.0000000
Done writing stats.... at dd−mmm−yyyy hh:mm:ss.01
```

Notice that the duplicity and null factors do not get computed at all.

The restriction on maximum cardinality at 249.95 million has been increased to approximately 2**52.

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.3.6 Incorrect LOCK_TIMEOUT With RMU /BACKUP /ONLINE /LOCK_TIMEOUT /LOG

Bug 4047148

In prior releases of Oracle Rdb, if both RMU/BACKUP/AFTER_JOURNAL and RMU/BACKUP specify LOCK_TIMEOUT then it appears that subsequent timeouts are increased. For example, if /LOCK_TIMEOUT=60 is specifed on an after−image journal backup, it stalls for approximately this time. If the database backup also has a 60 second wait time, then this appears to double and it waits for 2 minutes.

```
Session 1:

SQL> attach 'file mf_personnel';
SQL> update employees set sex='M'

Session 2:
$ set noon
$ show time
   3−DEC−2008 17:17:09
$ rmu/back/after/lock=60 mf_personnel  mf_per.aij
%RMU−W−DATACMIT, unjournaled changes made; database may not be recoverable
%RMU−F−TIMEOUT, timeout on quiet
−COSI−W−CANCEL, operation canceled
%RMU−F−FTL_BCK, Fatal error for BACKUP operation at  3−DEC−2008 17:18:11.91
$ show time
   3−DEC−2008 17:18:12
$ rmu/back/log/online/lock=60 mf_personnel mf_per.rbf
%RMU−I−QUIETPT, waiting for database quiet point at  3−DEC−2008 17:18:22.27
%RMU−F−TIMEOUT, timeout on quiet
−COSI−W−CANCEL, operation canceled
%RMU−F−FTL_BCK, Fatal error for BACKUP operation at  3−DEC−2008 17:20:22.15
$ show time
   3−DEC−2008 17:20:22
```

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.3.7 RMU /RESTORE /ONLINE /JUST_CORRUPT Bugcheck

Bug 8568870

In some cases, the "RMU /RESTORE /ONLINE /JUST_CORRUPT" command can fail with an internal consistency failure bugcheck "footprint" similar to:

```
Exception occurred at RMU72\RMUCLI$RESTORE + 000038A4
COSI−F−BUGCHECK, internal consistency failure
Called from RMU72\RMU_DISPATCH + 00000F44
Called from RMU72\RMU_STARTUP + 000004CC
Called from RMU72\RMU$MAIN + 00000034
```

This problem can be triggered by having deleted storage areas as in the following example:

```
$ SQL$
 CREATE DATABASE FILENAME TESTDB
    NUMBER OF CLUSTER NODES IS 1 NUMBER OF USERS IS 10
    CREATE STORAGE AREA A0 ALLOCATION IS 8 SNAPSHOT ALLOCATION IS 8
    CREATE STORAGE AREA A1 ALLOCATION IS 8 SNAPSHOT ALLOCATION IS 8
    CREATE STORAGE AREA A2 ALLOCATION IS 8 SNAPSHOT ALLOCATION IS 8
    CREATE STORAGE AREA A3 ALLOCATION IS 8 SNAPSHOT ALLOCATION IS 8
    CREATE STORAGE AREA A4 ALLOCATION IS 8 SNAPSHOT ALLOCATION IS 8
    CREATE STORAGE AREA A5 ALLOCATION IS 8 SNAPSHOT ALLOCATION IS 8
    CREATE STORAGE AREA A6 ALLOCATION IS 8 SNAPSHOT ALLOCATION IS 8
    CREATE STORAGE AREA A7 ALLOCATION IS 8 SNAPSHOT ALLOCATION IS 8
    CREATE STORAGE AREA A8 ALLOCATION IS 8 SNAPSHOT ALLOCATION IS 8
    CREATE STORAGE AREA A9 ALLOCATION IS 8 SNAPSHOT ALLOCATION IS 8;
 ALTER DATABASE FILENAME TESTDB
    DROP STORAGE AREA A3
    DROP STORAGE AREA A4
    DROP STORAGE AREA A5;
 CREATE TABLE T1 (C1 INTEGER, C2 CHAR(5));
 CREATE STORAGE MAP M1 FOR T1 STORE IN A1;
 CREATE TABLE T8 (C1 INTEGER, C2 CHAR(5));
 CREATE STORAGE MAP M8 FOR T8 STORE IN A8;
 INSERT INTO T1 VALUES (11, 'ONE') RETURNING DBKEY;
 INSERT INTO T8 VALUES (81, 'ONE') RETURNING DBKEY;
 COMMIT;
 EXIT;
$ RMU/BACKUP/NOLOG TESTDB TESTDB
$ RMU/ALTER TESTDB ! Invalidate checksums
    AREA 3 PAGE 5
    DEPOSIT CHECKSUM = 101010101
    COMMIT
    AREA 10 PAGE 5
    DEPOSIT CHECKSUM = 101010101
    COMMIT
    EXIT
$ RMU/OPEN/WAIT TESTDB
$ SQL$ ! Detect checksum errors and add to CPT
    ATTACH 'FILE TESTDB';
    SELECT * FROM T1;
    SELECT * FROM T8;
    EXIT;
$ RMU/SHOW CORRUPT TESTDB
$ RMU/RESTORE/ONLINE/JUST_CORRUPT/NOLOG TESTDB
$ RMU/CLOSE/WAIT TESTDB
```

As a workaround, omit the "/ONLINE" qualifier; perform the /JUST_CORRUPT restore operation offline.

This problem has been corrected in Oracle Rdb Release 7.2.4. The online /JUST_CORRUPT restore now correctly detects and ignores invalid FILID entries.

3.3.8 RMU /LOAD /PARALLEL COSI−F−READERR Without VMS BYPASS Privilege

Bug 8468385

The RMU/LOAD/PARALLEL documentation correctly states that the VMS BYPASS process privilege is not required for RMU/LOAD/PARALLEL as long as the Oracle Rdb database Access Control List permits the user to execute the RMU/LOAD command. However, a %COSI−F−READERR error resulted from a code problem that caused the VMS BYPASS process privilege to be required to access a VMS mailbox device used for communicating with the executor processes that are created by RMU/LOAD/PARALLEL. This problem has been fixed and the code now conforms to the documentation that the BYPASS process privilege is not required for RMU/LOAD/PARALLEL.

The following example shows the problem. If the user is not granted the VMS BYPASS privilege, a %COSI−F−READERR is returned when each executor is created by the RMU/LOAD/PARALLEL command when loading the TEST_LOAD table from the TEST_LOAD.UNL unload file in the TESTDB.RDB database. No data is loaded and the parallel load terminates with a fatal error status.

```
RYEROX>show process/privileges
Authorized privileges:
 CMKRNL       NETMBX       OPER         PRMGBL       SYSGBL       SYSLCK
 SYSPRV       TMPMBX       WORLD

Process privileges:
 NETMBX               may create network device
 TMPMBX               may create temporary mailbox

$rmu/show privileges testdb.rdb
    (IDENTIFIER=[*,*]ACCESS=READ+WRITE+CONTROL+RMU$ALL)

$rmu/unload testdb.rdb test_load test_load.unl
$rmu/load/debug=trace/commit=10000 −
/parallel=(exec=2,buff=64) testdb.rdb test_load test_load.unl

* Using EXECUTOR_IMAGE:"SYS$COMMON:[SYSLIB]RDMRLE72.EXE"

%COSI−F−READERR, read error

%COSI−F−READERR, read error

%RMU−I−DATRECREAD,  0 data records read from input file.
%RMU−I−DATRECSTO,   0 data records stored.
%RMU−F−FTL_LOAD, Fatal error for LOAD operation at 27−APR−2009 09:36:20.77
```

The only way to avoid this problem in earlier versions of Oracle Rdb which have this problem is to grant the VMS BYPASS process privilege to the account from which the parallel load is being run or to do a non−parallel load of the database table.

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.4 LogMiner Errors Fixed

3.4.1 Continuous LogMiner Startup Serialized With AIJ Backups

Bug 7634512

Although it is documented that it is not permitted to perform AIJ backup operations when using the Continuous LogMiner feature until the Continuous LogMiner has fully transitioned to the live AIJ files, doing so could result in unintended data loss in the output stream from the LogMiner.

The impact of this problem has been reduced in Oracle Rdb Release 7.2.4. The Continuous LogMiner now utilizes a global "AIJ Backup" lock to serialize the startup of the Continuous LogMiner with AIJ backup operations.

3.4.2 RMU /UNLOAD /AFTER_JOURNAL Allows /RESTART Without /CONTINUOUS

Previously, the /RESTART qualifier required the /CONTINUOUS qualifier for restarting the LogMiner. This restriction has been relaxed. The /RESTART qualifier is now allowed with or without the CONTINUOUS qualifier.

In addition, when the /RESTART qualifier is specified, it is no longer enforced that the first AIJ backup file supplied be from after a quiet−point AIJ backup. This implies that it is possible that the transaction returned with the specified AERCP (AIJ Extract Restart Control Point) could be an incomplete transaction (where not every modification made by the transaction is returned in the output data stream). Applications that use /RESTART are expected to understand this behavior and to ignore the transaction with the restarted AERCP.

This problem has been corrected in Oracle Rdb Release 7.2.4.

3.5 Row Cache Errors Fixed

3.5.1 Incremental Backup With Row Cache

Bug 8363084

Previously, it was possible for incremental database backups to not correctly save all modified database rows since the last full backup when the row cache feature was in use. When modified rows were copied from cache back to the database, the "MAX_SNAP_TSN" field on the database page was not correctly maintained. This field is used by the incremental database backup feature as an indication of when rows on the page may have been modified and if it is out of date, the incremental backup may not consider the page content as a candidate to be saved.

As a possible workaround for this problem, perform full database backups rather than incremental database backups when using the row cache feature.

This problem has been corrected in Oracle Rdb Release 7.2.4. The "MAX_SNAP_TSN" field on the database page is now maintained correctly when modified rows are copied from cache back to the database.

3.6 Hot Standby Errors Fixed

3.6.1 Hot Standby LRS Database Prefetch Count Limit

Previously, the LRS process would set its APF (asynchronous prefetch) depth to half of its buffer count. For large numbers of buffers, this depth could result in a significant number of outstanding database prefetch IO requests. In some cases, this could result in possible quota exhaustion or storage controller saturation.

This problem has been addressed by providing additional control of the LRS process APF depth along with a lower default limit.

The logical name (must be defined system−wide prior to the startup of an LRS process) RDM$BIND_LRS_MAX_APF_DEPTH can be used to limit the maximum number of default LRS APF IO depth. If not specified, the default value is 500. The minimum value is 2 and the maximum value is 524288. If the database specifies a higher APF depth value, that value will be utilized.

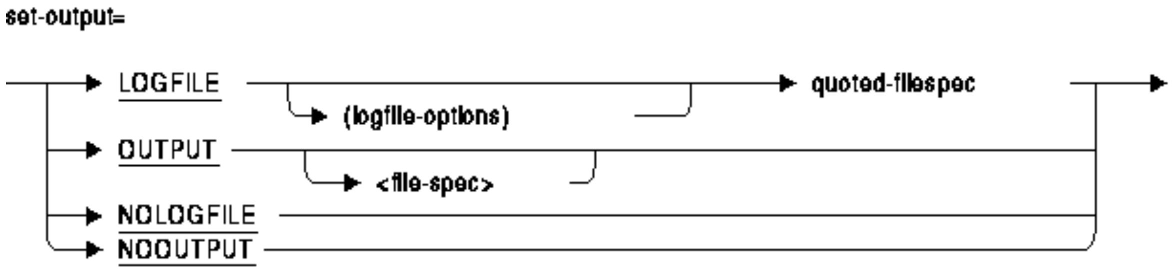
Chapter 4
Enhancements And Changes Provided in Oracle Rdb Release 7.2.4.1

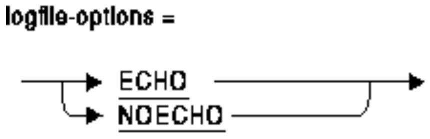
4.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.4.1

4.1.1 New SET LOGFILE Command

This release of Oracle Rdb adds a new SET LOGFILE statement to interactive SQL. This statement allows the executing SQL script to save output to an OpenVMS file.

Syntax





Arguments

- quoted−filespec
  A valid OpenVMS file specification. Output from interactive SQL will be written to this file.
- NOLOGFILE
  Closes the current output file specified by a prior SET LOGFILE (or SET OUTPUT command).
- NOOUTPUT
  Suspends writing to the output file.
- ECHO
  In addition to writing the output to the designated file, all commands and errors generated by interactive SQL are also written to SYS$OUTPUT.
- NOECHO
  Disable output to SYS$OUTPUT. All commands and errors generated by interactive SQL are only written to the output file.

Usage Notes

- SET LOGFILE is functionally equivalent to the SET OUTPUT statement. However, the SET OUTPUT statement is limited in functionality and is maintained for backward compatibility.

- Files opened with SET OUTPUT and SET LOGFILE can be subsequently processed by either a SET NOOUTPUT or a SET NOLOGFILE command.
- A SET LOGFILE command that does not specify a file is equivalent to SET NOLOGFILE.
- Output written by external functions, SQL TRACE statements, and other output enabled by the SET FLAGS command is never written to the SQL log file. Therefore, it cannot be captured using the statement.

Examples

Saving the output from a script

The following example shows the use of SET LOGFILE to save the output from a script without echoing the results.

1. The script being executed.

   ```
   set verify;
   start transaction read only;
   set logfile (noecho) 'saved_date.log';
   select rdb$flags from rdb$database;
   set nologfile;
   show alias;
   rollback;
   ```
2. The output as seen during the Interactive SQL session.

   ```
   SQL> start transaction read only;
   SQL>
   SQL> set logfile (noecho) 'saved_date.log';
   SQL>
   SQL> show alias;
   Default alias:
       Oracle Rdb database in file SQL$DATABASE
   SQL> rollback;
   ```
3. The output saved in the log file.

   ```
   SQL>
   SQL> select rdb$flags from rdb$database;
      RDB$FLAGS
              0
   1 row selected
   SQL>
   SQL> set nologfile;
   ```

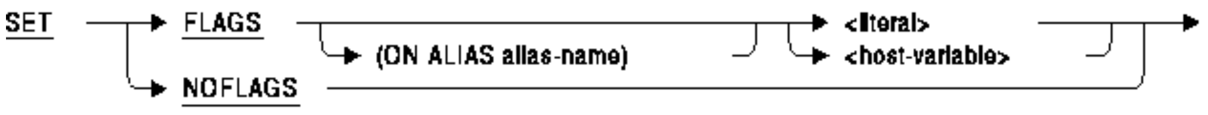
4.1.2 SET FLAGS Statement Now Allows ON ALIAS Clause

This release of Oracle Rdb extends the SET FLAGS statement to support an ON ALIAS clause. The default behavior for SET FLAGS is to establish the flag settings on all currently attached databases. This new clause will allow the database administrator to set flags on just one database alias.

The following example shows a case where enabling AUTO_OVERRIDE required DBADM privilege on the target database but not on the source database. It may be that the current user does not have (or really need) DBADM privilege on that database.

```
SQL> -- Now enable AUTO_OVERRIDE on only one database
SQL> set flags (on alias abc_a) 'auto_override';
SQL> set flags (on alias abc_b) 'none';
SQL> insert into abc_a.SAMPLE_TABLE select * from abc_b.SAMPLE_SOURCE;
SQL> commit;
```

Syntax



Arguments

- alias−name
  The name of an alias as declared by the ATTACH or CONNECT statement. If no ALIAS clause is used, then the alias name will default to RDB$DBHANDLE.

4.1.3 SQL Compiler−Generated Name Uniqueness Enhanced

Bug 4119771

In previous versions of Oracle Rdb, the SQL module language compiler (SQLMOD) and the SQL precompiler (SQLPRE) would generate object names based solely on the system time. This could, in some cases, result in duplicate names being generated for multiple different objects that were compiled at the same time by different processes.

This problem, for example, could cause linker warnings that show the symbol with a name similar in format to "SQL$PROC_1_A3DB62_3331BC" that had been created twice from within different object modules.

This problem has been corrected in Oracle Rdb Release 7.2.4.1. The SQL module language compiler and the SQL precompiler now create unique names within a system or a cluster. The names are comprised of a prefix, a request number and a string comprised of a cluster−wide unique value. The unique value includes components of the system time and the ID of the compiling process. The format of the new names is similar to "SQL$PRC9_DJHS2IHDBAA1G8BQ26A0".

4.1.4 Reduced CPU Usage and Improved Performance

Several performance enhancements have been implemented in this release of Oracle Rdb. Most of these changes are either specific to applications running on I64 systems or will have a greater effect on I64 systems. These enhancements include:

- Streamlined code sequences
- Reduced alignment faults

4.1.5 New RMU /SHOW STATISTICS /WRITE_REPORT_DELAY=n Feature

Bug 3199615

Previously, it was not possible to use the RMU /SHOW STATISTICS to write a report file in non−interactive mode.

This problem has been corrected in Oracle Rdb Release 7.2.4.1. The /WRITE_REPORT_DELAY=n qualifier specifies that statistics are to be collected for "n" seconds (default of 60 seconds) and then a report file written and then the RMU /SHOW STATISTICS utility will exit. /WRITE_REPORT_DELAY implies /NOINTERACTIVE.

Chapter 5
Enhancements And Changes Provided in Oracle Rdb Release 7.2.4

5.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.4

5.1.1 Date/Time Arithmetic Enhancements

Bug 6219485

This release of Oracle Rdb lifts many restrictions on the DATE VMS data type. SQL now treats DATE VMS type as a TIMESTAMP(2) for the purposes of the add and subtract with intervals or when generating intervals.

- A year−month interval can be added to or subtracted from a DATE VMS value and result in a DATE VMS value.
- A day−time interval can be added to or subtracted from a DATE VMS value and result in a DATE VMS value.
- A DATE VMS value can be subtracted from another DATE VMS value to produce either a year−month interval or a day−time interval.
- A DATE ANSI value can be subtracted from a DATE VMS value to produce a year−month interval.
- A DATE VMS value can be subtracted from a DATE ANSI value to produce either a year−month interval or a day−time interval.
- A DATE VMS value can be subtracted from a TIMESTAMP value to produce either a year−month interval or a day−time interval.
- A TIMESTAMP value can be subtracted from a DATE VMS value to produce either a year−month interval or a day−time interval.

Also in this release the following rules have been relaxed.

- Relax rule that TIME values could only be subtracted if they had the same fractional seconds precision.
- Relax rules that TIMESTAMP values could only be subtracted if they had the same fractional seconds precision.
- Relax assignment rules and let Rdb handle truncating time portion when TIMESTAMP is assigned to a DATE ANSI column, variable or parameter.
- Add rule that merge of DATE VMS and TIMESTAMP(n) will always be TIMESTAMP(n), where n is inherited from the TIEMSTAMP expression. Merge rules are used by UNION, INTERSECT, EXCEPT, MINUS operators and CASE expression processing.

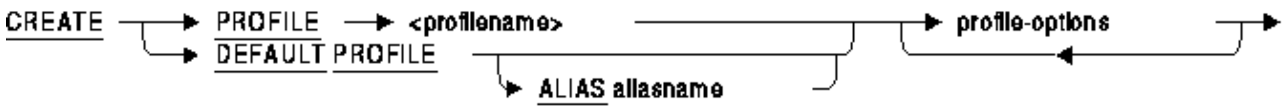
These enhancements have been made in Oracle Rdb Release 7.2.4.

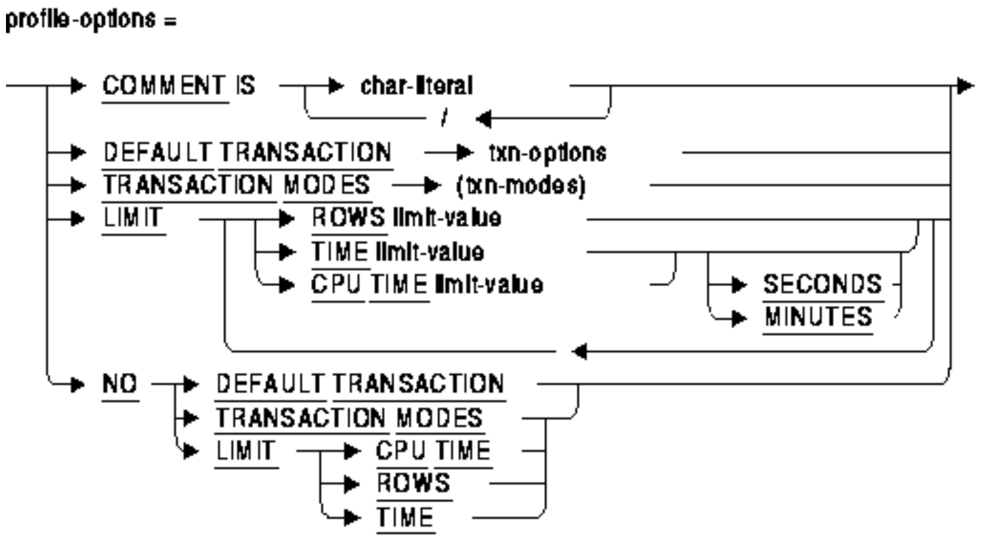
5.1.2 New DEFAULT PROFILE Feature

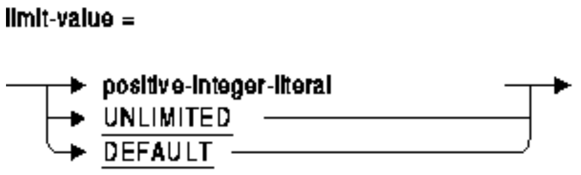
This release of Oracle Rdb enhances the PROFILE support with a new DEFAULT profile. When a user attaches to the database using ATTACH, CONNECT or SET SESSION AUTHORIZATION, they will either load their assigned profile definition or inherit the default profile (if defined).

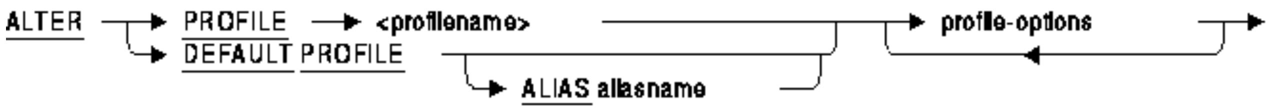
Syntax

The CREATE, ALTER and DROP PROFILE syntax is changed as shown. The existing profile−options diagram remains the same.

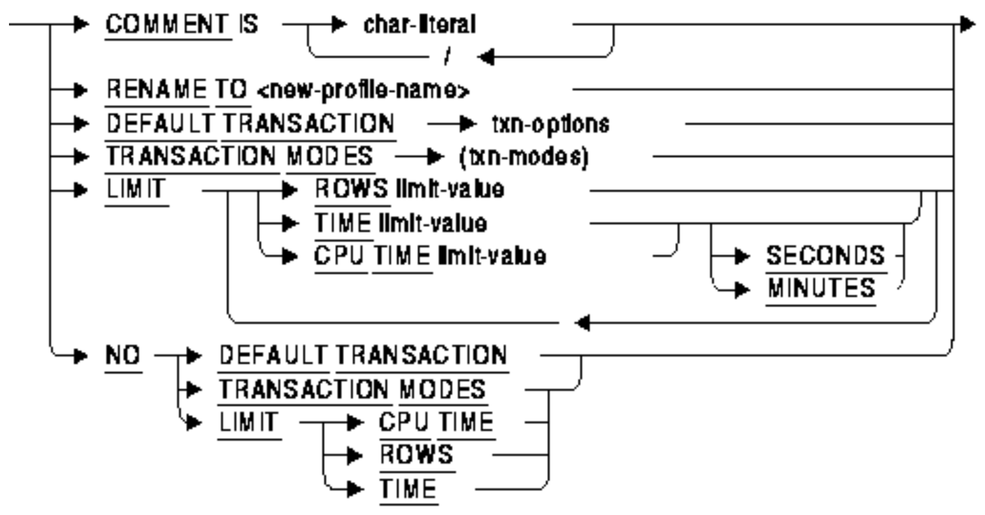


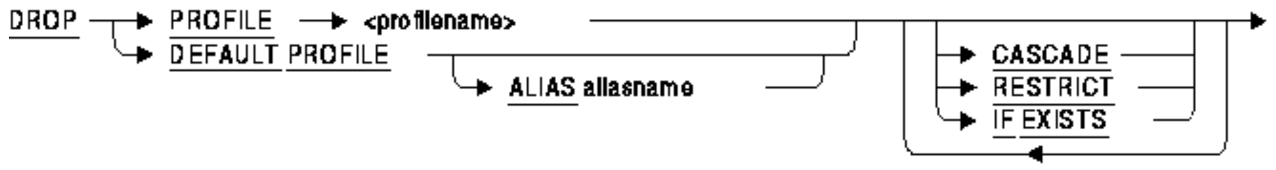






profile-options =





Arguments

- ALIAS aliasname
  When attached to multiple databases, the aliasname is required to direct the CREATE, ALTER or DROP command to the appropriate database.
- DEFAULT PROFILE
  Creates the special profile RDB$DEFAULT_PROFILE. This profile will be used by any user who is not assigned a profile using the PROFILE clause of CREATE or ALTER PROFILE.

Usage Notes

- It is possible to restrict the transaction modes to READ ONLY using the default profile. Use caution in this case because it is possible that no user will have READ WRITE access to undo such a definition. In this case, you can define the logical name RDMS$SET_FLAGS to the value PROFILE_OVERRIDE to allow a suitably privileged user to start a transaction without using the transaction mode restrictions in the default profile. Such a user must have database SECURITY privilege, possibly inherited from the OpenVMS SECURITY process privilege.

5.1.3 RMU /DUMP/BACKUP/OPTIONS=ROOT /HEADER_ONLY Displays the Header Information Only

Bug 8235615

A new feature has been added to RMU/DUMP/BACKUP/OPTIONS=ROOT command to process only the header information when the /HEADER_ONLY qualifier is used.

In prior releases of Oracle Rdb, the user had to wait until the entire backup file (.RBF) was processed. If the backup file was stored on tape and spanned multiple tapes then all the tapes had to be mounted and processed. When using /HEADER_ONLY, RMU now ceases processing of the backup file once the header has been displayed.

```
$ RMU/DUMP /BACKUP/OPTIONS=ROOT /HEADER_ONLY DBNODE$LMA200:GLORY.RBF
*------------------------------------------------------------------------------
* Oracle Rdb V7.2-4                                    26-FEB-2009 07:21:58.69
*
* Dump of Database Backup Header
*     Backup filename: GLORY.RBF
*     Backup file database version: 7.2
*
*------------------------------------------------------------------------------

Database Parameters:
    Root filename is "USER1:[BUG.8235615.FIX]GLORY.RDB;1"
    Created at 25-APR-2007 16:43:05.52
    Oracle Rdb structure level is 72.1
    Maximum user count is 23
    Maximum node count is 3
    Database open mode is AUTOMATIC
    Database close mode is AUTOMATIC
    Database will be mapped in process space
    All transaction modes are allowed
    Prestarted transactions are enabled
    Snapshot mode is NON-DEFERRED
    Statistics are enabled
    Operator notification is disabled
    Logical area count is 512
    Storage Areas...
      - Active storage area count is 4
      - Reserved storage area count is 8
    Row Caches...
      - Active row cache count is 0
      - Reserved row cache count is 1
      - Checkpoint information
          No time interval is specified
          Default source is updated rows
          Default target is backing file
          Default backing file directory is database directory
          RUJ Global Buffers are disabled
      - WARNING: Maximum node count is 3 instead of 1
      - WARNING: After-image journaling is disabled
      - WARNING: Fast commit is disabled
    Buffers...
      - Default user buffer count is 2000
      - Default recovery buffer count is 2000 (stored as 20)
      - Global buffers are disabled
          Global buffer count is 115
          Maximum global buffer count per user is 5
          Large memory is disabled
      - Buffer size is 12 blocks
          Maximum pages per buffer is 6
      - Asynchronous pre-fetch is enabled
          Maximum pre-fetch depth is 8 buffers
      - Detected asynchronous pre-fetch is enabled
          Maximum pre-fetch depth is 4 buffers
          Pre-fetch threshold is 4 buffers
      - Asynchronous batch-write is enabled
          Clean buffer count is 5
          Maximum batch size is 10 buffers
      - Optimized page transfer is disabled
    Locking...
      - Adjustable record locking is enabled
          Fanout factor 1 is 10 (10 pages)
          Fanout factor 2 is 10 (100 pages)
          Fanout factor 3 is 10 (1000 pages)
      - Carry-over lock optimization is enabled
      - Lock tree partitioning is disabled
    RUJ Journaling...
      - No default recovery-unit journal directory
    AIJ Journaling...
      - After-image journaling is disabled
      - Database is configured for 7 journals
      - Reserved journal count is 7
      - Available journal count is 0
      - LogMiner is disabled
      - 7 journals can be created while database is active
      - Shutdown time is 60 minutes
      - Backup operation is manual
      - Default backup filename edits are not used
      - Log server startup is MANUAL
      - Journal overwrite is disabled
      - AIJ cache on "electronic disk" is disabled
      - Default journal allocation is 512 blocks
      - Default journal extension is 512 blocks
      - Default journal initialization is 512 blocks
      - Current roll-forward sequence number is 0
      - Current backup sequence number is 0
    Fast Commit...
      - Fast commit is disabled
      - No checkpointing AIJ interval is specified
      - No checkpointing time interval is specified
      - No checkpointing transaction interval is specified
      - Commit to AIJ optimization is disabled
      - Transaction interval is 256
    Hot Standby...
      - WARNING: After-image journaling is disabled
      - WARNING: Fast commit is disabled
      - WARNING: Log server startup is MANUAL
      - Informational: Operator notification is disabled
      - Database is not currently being replicated
    Security Auditing...
      - Security auditing is disabled
      - Security alarm is disabled
      - No audit journal filename is specified
      - No alarm name is specified
      - Synchronous audit record flushing is disabled
      - Audit every access
    Database Backup...
      - Fast incremental backup is enabled
      − Last full database backup was on 26−FEB−2009 07:16:56.09
      − Full database backup TSN is 0:128
      − Database was restored on 26−FEB−2009 06:52:11.82
    Derived Data...
      − Global section size
          With global buffers disabled is 277187 bytes (1MB)
          With global buffers enabled is 1036584 bytes (1MB)
            With Large memory global buffers enabled...
              Database TROOT section is 330024 bytes (1MB)
              Large memory global buffers section is 706560 bytes (1MB)
      − Row Cache RUJ buffers section size is 6041088 bytes (6MB)

Database root file ACL

(IDENTIFIER=[RDB,SFRANN],ACCESS=READ+WRITE+CONTROL+RMU$ALTER+RMU$ANALYZE+
     RMU$BACKUP+RMU$CONVERT+RMU$COPY+RMU$DUMP+RMU$LOAD+
     RMU$MOVE+RMU$OPEN+RMU$RESTORE+RMU$SECURITY+RMU$SHOW+RMU$UNLOAD+RMU$VERIFY)
```

5.1.4 GET ENVIRONMENT Now Supports SQLCODE and SQLSTATE Capture

This release of Oracle Rdb allows the GET ENVIRONMENT (SESSION) command to return the SQLCODE and SQLSTATE of the last executed statement. The keyword SQLCODE returns an INTEGER value and SQLSTATE returns a CHAR(5) value. The execution of the GET ENVIRONMENT statement will clear these values so both should be fetched in the same statement.

The following example shows a raised error and the use of GET ENVIRONMENT to capture the SQLCODE and SQLSTATE.

```
SQL> declare :st char(5);
SQL> declare :sc integer = −1;
SQL>
SQL> begin set :sc = :sc / 0; end;
%RDB−E−ARITH_EXCEPT, truncation of a numeric value at runtime
−COSI−F−ARITH, arithmetic exception
−COSI−F−FLTDIV, floating point division exception
SQL>
SQL> get environment (session) :st = SQLSTATE, :sc = SQLCODE;
SQL>
SQL> print :st, :sc;
 ST              SC
 22003          −304
SQL>
```

Once the SQLCODE or SQLSTATE value has been saved to a declared variable, it can be used to conditionally execute a COMMIT or a ROLLBACK.

```
begin
if :sc < 0 then
    rollback;
else
    commit;
end if;
end;
```

5.1.5 Timestamp Added to Messages For RMU LOAD and UNLOAD

In order to help judge progress of RMU LOAD and UNLOAD operations, a timestamp has been added to the RMU−I−DATRECSTO and RMU−I−DATRECUNL progress messages. The following example shows these timestamps.

```
$RMU /UNLOAD MFP C1 C1
%RMU−I−DATRECUNL,   200000 data records unloaded  5−JUN−2009 07:58:17.23.
$RMU /LOAD /COMMIT=75000 /LOG_COMMIT MFP C1 C1
%RMU−I−DATRECSTO,   75000 data records stored  5−JUN−2009 07:58:43.09.
%RMU−I−DATRECSTO,   150000 data records stored  5−JUN−2009 07:58:43.12.
%RMU−I−DATRECSTO,   200000 data records stored  5−JUN−2009 07:58:43.14.
```

5.1.6 New SET SQLDA Statement

Bugs 1088554, 4179408, 5414051, and 7022262

This release of Oracle Rdb introduces a new SET SQLDA statement.

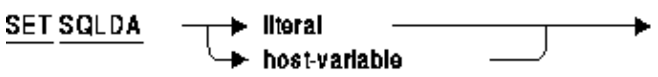
The SQLDA Statement allows a programmer using Dynamic SQL to alter the way the SQLDA (and SQLDA2) and Dynamic SQL statements are processed by Oracle Rdb.

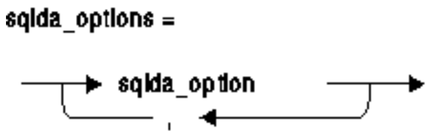
Environment

You can use the SET SQLDA statement:

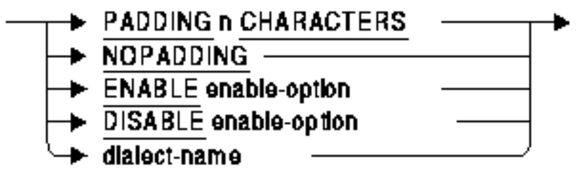
- In dynamic SQL as a statement to be dynamically executed

Syntax

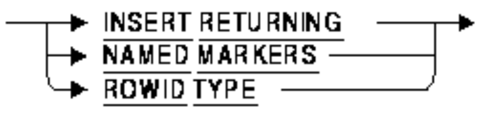




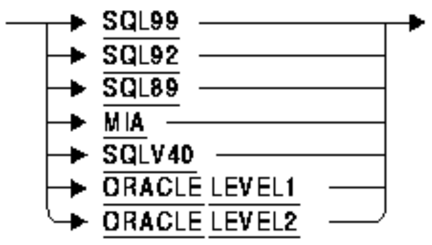
sqlda_option =



enable-option =



dialect-name =



Arguments

- literal
  host−variable
  The parameter passed to the statement must be a literal or a host variable containing one or more SQLDA options (see sqlda_options syntax diagram for details). If more than one option is specified, they must be separated by commas.
- sqlda_options
  One or more keyword clauses. If more than one clause is specified, they must be separated by commas.
- ENABLE
  The ENABLE clause activates one of the following behaviors for Dynamic SQL.
    - ♦ INSERT RETURNING − The default behavior of INSERT ... RETURNING when executed by dynamic SQL is to place parameters from the RETURNING INTO clause into the INPUT SQLDA. This behavior is maintained for backward compatibility. This option allows the programmer to force different (and correct) behavior for the non−compound use of this statement.

Note

If the INSERT RETURNING statement is included in a compound statement, the parameters are handled correctly.

- ♦ NAMED MARKERS − as well as traditional parameters markers (?). Dynamic SQL will now accept named, host variable style parameter markers. See the Usage Notes for further details and examples.
- ♦ ROWID TYPE − returns DBKEY values as a special type (SQLDA_ROWID, 455) to make processing of the DBKEY values easier. For instance, in prior releases, the SQLDA name field (SQLNAME) for DBKEY entries in the SQLDA was the only way to distinguish these values from other CHAR or VARCHAR columns − it would be either DBKEY or ROWID. If a query renamed the DBKEY column, then the application had no information in the SQLDA to indicate that the CHAR or VARCHAR value was binary data. In all respects, the SQLDA_ROWID type appears as a fixed length string of octets (possibly containing octets of zero which the C language would treat as a NULL terminator for a string).

- DISABLE
  The DISABLE clause deactivates one of the specified behaviors for Dynamic SQL. See ENABLE clause for a list of options.
- ORACLE LEVEL1
  ORACLE LEVEL2
  Either of these options will set the SQLDA to supply enhanced semantics. These options are currently reserved for use of the OCI Services for Rdb product that is part of the Oracle Rdb SQL/Services component. This setting also implicitly enables NAMED MARKERS.
- PADDING n CHARACTERS
  This option directs SQL to configure the SQLDA with larger CHARACTER VARYING strings than would normally be seen. The value of n is an unsigned numeric literal that specifies the number of characters that are added to the estimated length. Any CHARACTER (CHAR) types are converted to CHARACTER VARYING (VARCHAR). This rule is applied to comparison operators <, <=, >, >=, =, <>, and string functions (STARTING WITH, CONTAINING).
- NOPADDING
  This option sets the number of padding characters to 0. This also implies that derived CHARACTER (CHAR) types are not converted to CHARACTER VARYING (VARCHAR) when PADDING CHARACTERS is used.

  Note

  Oracle recommends that applications always check for SQLDA_CHAR and SQLDA_VARCHAR so that the correctly formatted data is made available to SQL.

  This is the default setting.
- SQL99
  SQL92
  MIA
  SQL89
  SQLV40
  Any of these options will revert to the default semantic for the SQLDA which includes disabling NAMED MARKERS.

Usage Notes

- The ORACLE LEVEL1 and ORACLE LEVEL2 settings are reserved for use by Oracle Corporation. Current behavior of this setting may change with any given release based on requirements of the OCI Services for Rdb component. This setting changes the usage of various SQLDA and SQLDA2 fields.
- Keywords may not be abbreviated and the clauses must be fully specified.
- The SET DIALECT command will implicitly enable NAMED MARKERS if the dialect is changed to either ORACLE LEVEL1 or ORACLE LEVEL2.
- The SET DIALECT command will implicitly disable NAMED MARKERS if the dialect is changed to any dialect other than ORACLE LEVEL1 or ORACLE LEVEL2.
- When NAMED MARKERS are enabled, the contents of the SQLDA and SQLDA2 will reflect one entry for each name. When traditional parameter markers are used, a SQLDA (or SQLDA2) entry will exist for each marker (?) encountered. This change in behavior can simplify the query encoding as well lead to more efficient strategy creation.

Examples

Using the NAMED MARKERS Feature

This example shows that enabling the NAMED MARKERS feature will allow SQL to prompt for one value and the displayed Rdb strategy shows that only one variable is used.

```
-> SET SQLDA 'ENABLE NAMED MARKERS';
-> SELECT LAST_NAME FROM EMPLOYEES WHERE FIRST_NAME = :F_NAME AND LAST_NAME <>
:F_NAME;
in:  [0] typ=449 len=46
out: [0] typ=453 len=14
[SQLDA - reading 1 fields]
-> Alvin
Tables:
  0 = EMPLOYEES
Conjunct: (0.FIRST_NAME = <var0> AND (0.LAST_NAME <> <var0>))
Get     Retrieval sequentially of relation 0:EMPLOYEES
 0/FIRST_NAME/Varchar(42/46): Alvin
[SQLDA - displaying 1 fields]
 0/LAST_NAME: Toliver
[SQLDA - displaying 1 fields]
 0/LAST_NAME: Dement
```

Using the PADDING Feature

The following example shows that the derived type for the named parameter MI is a SQLDA_CHAR (453) of length 1. The input data ('AA') is truncated on assignment and the incorrect results are returned. By adding a small padding, the type is changed to SQLDA_VARCHAR (449) of length 3 and a correct comparison is performed.

```
-> ATTACH 'filename sql$database';
-> SET SQLDA 'enable named markers, nopadding';
-> SELECT LAST_NAME FROM EMPLOYEES WHERE MIDDLE_INITIAL = :MI;
in:  [0] typ=453 len=1
out: [0] typ=449 len=18
[SQLDA - reading 1 fields]
-> AA
[SQLDA - displaying 1 fields]
 0/LAST_NAME: Toliver
[SQLDA - displaying 1 fields]
 0/LAST_NAME: Lengyel
[SQLDA - displaying 1 fields]
```

```
 0/LAST_NAME: Robinson
[SQLDA − displaying 1 fields]
 0/LAST_NAME: Ames
−> SET SQLDA 'padding 2 characters';
−> SELECT LAST_NAME FROM EMPLOYEES WHERE MIDDLE_INITIAL = :MI;
in:  [0] typ=449 len=7
out: [0] typ=449 len=18
[SQLDA − reading 1 fields]
−> AA
−> EXIT;
Enter statement:
```

Note that the VARCHAR requires an extra 4 bytes for the length information in the SQLDA2 used by the Dynamic SQL testing program.

5.1.7 RMU /SHOW VERSION Displays System Architecture and Version

The RMU /SHOW VERSION command has been enhanced to include information about the system architecture and OpenVMS version as shown in the following example:

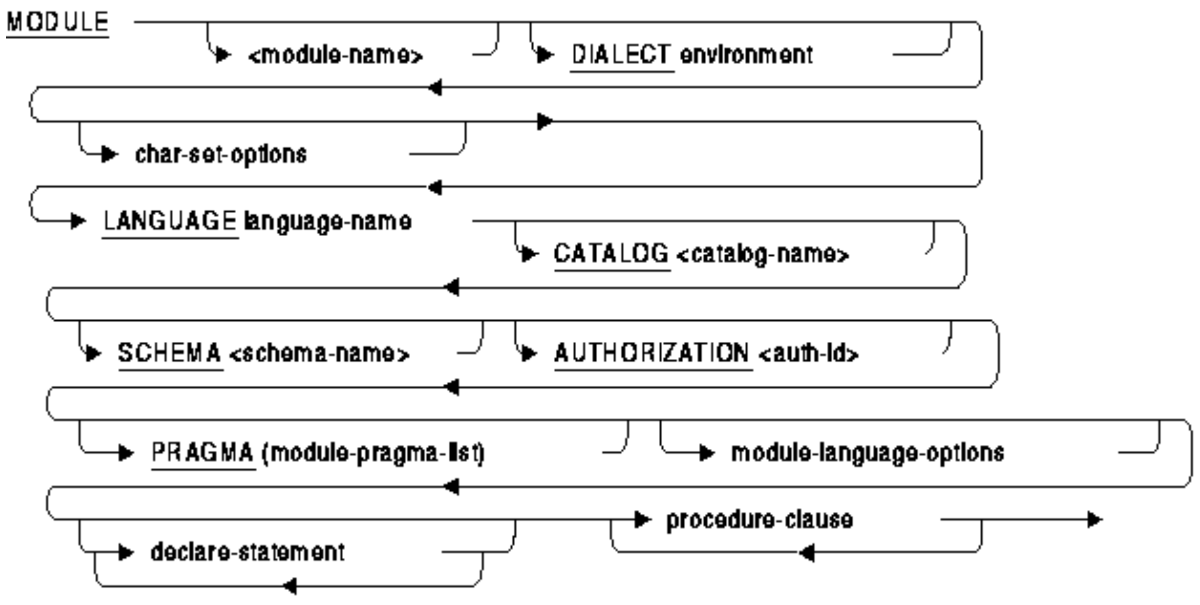
```
$ RMU /SHOW VERSION
Executing RMU for Oracle Rdb V7.2−400 on OpenVMS IA64 V8.3−1H1
$
```

5.1.8 New IDENT Option for SQL Module Language PRAGMA Clause

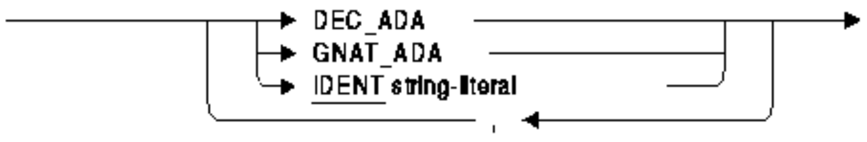
Many OpenVMS compilers allow the programmer to specify the object file IDENT string. This allows tracking of the correct version in linker map files (.map) and within object libraries using the LIBRARIAN command. This release of Oracle Rdb supports setting the IDENT string for the SQL module language.

The IDENT string is specified as part of the PRAGMA clause in the module header.

Syntax



module-pragma-list =



Usage Notes

- By using the PRAGMA clause with IDENT you can record an identification string in the object module generated by the SQL Module language. This IDENT string is recorded by the OpenVMS LINKER in the image itself and can be viewed in the generated MAP file, examined using ANALYZE/OBJECT, and by the LIBRARIAN command when the object module is stored in an object library.
- OpenVMS limits the IDENT string to a 15 octet string. If the string is longer than this (even with trailing spaces) then an error will be reported by the SQL Module Language compiler.
- If the IDENT clause is omitted, then the default version string will default to 'V1.0' as is the practice with many OpenVMS compilers. Prior versions of Oracle Rdb on Integrity systems would only provide the string '0'.

```
Module name:                                   "MODSQL$TEST"
Module version:                                "0"
Creation date/time:                            "15−JUN−2009 20:13"
Language name:                                 "Oracle Rdb SQL V7.2−351"
```

Examples

The following example shows the use of a PRAGMA clause in a module header to specify the module ident string.

Example 5−1 PRAGMA Clause in the Module Header

```
MODULE         MODSQL$TEST
DIALECT        SQL99
LANGUAGE       C
AUTHORIZATION  SAMPLE_USER
PRAGMA         (IDENT 'V1.2−300')
ALIAS          RDB$DBHANDLE
PARAMETER      COLONS
```

The DCL command ANALYZE/OBJECT can be used to examine the ident string in the object file.

Example 5−2 Examining the IDENT in the Object Module

```
$ sql$mod TEST
$ analyze/object TEST/interactive
This is an OpenVMS IA64 (Elf format) object file

Module Identification Information, in note section 2.

    Module name:                                "MODSQL$TEST"
    Module version:                             "V1.2−300"
    Creation date/time:                         "15−JUN−2009 20:04"
    Language name:                              "Oracle Rdb SQL V7.2−401"
Press RETURN to continue, or enter a period (.) for next file:
<Ctrl/Z>
$
```

Here is similar output from an OpenVMS Alpha system.

```
$ sql$mod TEST
$ analyze/object TEST
   .
   .
   .
This is an OpenVMS Alpha object file

1.  MODULE HEADER (EOBJ$C_EMH), 71 bytes

        structure level: 2
        maximum record size: 4088
        module name: "MODSQL$TEST"
        module version: "V1.0"
        creation   date/time: 16−JUN−2009 11:02
   .
   .
   .
```

This example shows the use of the LIBRARIAN to display the ident strings for object modules in a project object library.

```
$ librarian/list/full project.olb
Directory of ALPHA OBJECT library DISK1:[TESTER]PROJECT.OLB;1 on 16−JUN−2009
11:07:23
```

```
Creation date:  16-JUN-2009 11:07:11      Creator:  Librarian A09-30
Revision date:  16-JUN-2009 11:07:11      Library format:   3.0
Number of modules:      1                 Max. key length:  128
Other entries:          5                 Preallocated index blocks:    213
Recoverable deleted blocks:        0      Total index blocks used:        2
Max. Number history records:      20      Library history records:        0

MODSQL$TEST      Ident V1.2-300         Inserted 16-JUN-2009 11:07:11 5 symbols
```

5.1.9 New Keyword for SQL Module Language /PRAGMA Qualifier

This release of Oracle Rdb adds a new option to the /PRAGMA qualifier. The keyword IDENT can be used to pass a text string to the SQL Module Language compiler to be written to the Object Module Header.

The following example demonstrates the use of the qualifier to establish the generation of the compiler module.

```
$ SQL$MOD TEST/PRAGMA=IDENT="v1.2-32"
```

Usage Notes

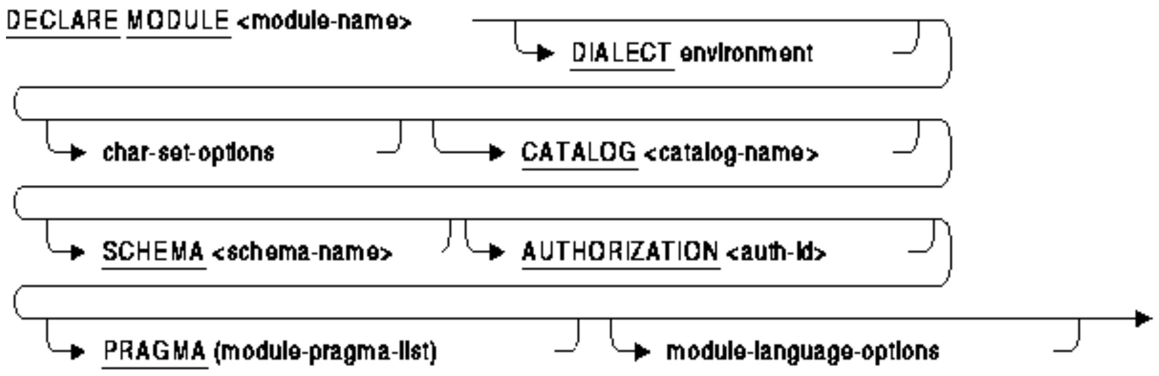
- If the PRAGMA (IDENT ...) clause is used as part of the MODULE header then that value will override any value used on the command line.
- The ANALYZE/OBJECT and LIBRARY commands can be used to display this IDENT string and the value will be displayed in LINKER map files.
- OpenVMS limits the IDENT string to a 15 octet string. If the string is longer than this (even with trailing spaces) then an error will be reported by the SQL Module Language compiler.

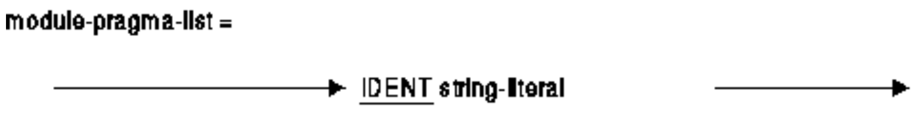
5.1.10 New IDENT Option for SQL Precompiler DECLARE MODULE Statement

Many OpenVMS compilers allow the programmer to specify the object file IDENT string. This allows tracking of the correct version in linker map files (.map) and within object libraries using the LIBRARIAN command. This release of Oracle Rdb supports setting the IDENT string for the SQL precompiler module header.

The IDENT string is specified as part of the PRAGMA clause in the module header.

Syntax





Usage Notes

- By using the PRAGMA clause with IDENT, you can record an identification string in the object module generated by the SQL Precompiler. This IDENT string is recorded by the OpenVMS LINKER in the image itself and can be viewed in the generated MAP file, examined using ANALYZE/OBJECT, and by the LIBRARIAN command when the object module is stored in an object library.
- OpenVMS limits the IDENT string to a 15 octet string. If the string is longer than this (even with trailing spaces), then an error will be reported by the SQL Precompiler.
- If the IDENT clause is omitted, then the default version string will default to 'V1.0' as is the practice with many OpenVMS compilers. Prior versions of Oracle Rdb on Integrity systems would only provide the string '0'.

```
    Module name:                                    "MODSQL$TEST"
    Module version:                                 "0"
    Creation date/time:                             "15−JUN−2009 20:13"
    Language name:                                  "Oracle Rdb SQL V7.2−351"
```

Examples

The following example shows the use of a PRAGMA clause in a module header to specify the module ident string.

Example 5−3 PRAGMA clause in the DECLARE MODULE statement

```
$ CREATE TEST_HDR.SQL
DECLARE
    MODULE         MODSQL$TEST
    DIALECT        SQL99
    AUTHORIZATION  SAMPLE_USER
    PRAGMA         (IDENT 'V1.2−300')
```

```
;
```

The DCL command ANALYZE/OBJECT can be used to examine the ident string in the object file. Note that the SQL Precompiler generates two object files which are concatenated. Therefore, the ANALYZE will show two MODULE HEADER records, one for the host language (C for example) and one from the SQL Precompiler.

Example 5−4 Examining the IDENT in the object module

```
$ sql$pre/CC TEST TEST_HDR
$ analyze/object TEST/output=SYS$OUTPUT
   .
   .
   .
This is an OpenVMS Alpha object file

175.  MODULE HEADER (EOBJ$C_EMH), 75 bytes
        structure level: 2
        maximum record size: 4088
        module name: "MODSQL$TEST"
        module version: "V1.2−300"
        creation   date/time: 16−JUL−2009 16:50
   .
   .
   .
```

5.1.11 New Keyword for SQL Precompiler PRAGMA Option

This release of Oracle Rdb adds a new keyword to the SQLOPTIONS qualifiers PRAGMA option. The keyword IDENT can be used to pass a text string to the SQL Precompiler to be written to the Object Module Header.

The following example demonstrates the use of the qualifier to establish the generation of the compiler module.

```
$ SQL$PRE/CC TEST/SQLOPTION=(PRAGMA=IDENT="v1.2−32")
```

Usage Notes

- If the PRAGMA (IDENT ...) clause is used as part of the DECLARE MODULE statement, then that value will override any value used on the command line.
- The ANALYZE/OBJECT and LIBRARY commands can be used to display this IDENT string and the value will be displayed in LINKER map files.
- OpenVMS limits the IDENT string to a 15 octet string. If the string is longer than this (even with trailing spaces) then an error will be reported by the SQL precompiler.

5.1.12 RDB_STATS_DATABASE Example Program

Accessing performance information in a tabular fashion for Oracle Rdb databases can often be beneficial. In particular, stored RMU /SHOW STATISTICS rate information in a database can be utilized to do trend anaysis and historical review of performance indicators.

RDB_STATS_DATABASE is a sample program that reads an RMU /SHOW STATISTICS binary file and converts all statistic values for each sample into a current rate per second. The statistics values are written to a database table named RMU$STATISTICS. If the RMU$STATISTICS table does not exist in the database, it will be created.

To use the RDB_STATS_DATABASE program, create a foreign command symbol with a value of "$SQL$SAMPLE:RDB_STATS_DATABASExx.EXE" (where xx is the version of Rdb) and pass an output database and an input binary statistics file name. The following example command sequence demonstrates one possible way that statistics can be gathered for one hour and then formatted.

```
$ RDB_STATS_DATABASE := $SQL$SAMPLE:RDB_STATS_DATABASE72
$ RMU /SHOW STATISTICS MFP -
        /NOINTERACTIVE -
        /OUTPUT = 2008-11-16-00-56.STATS -
        /UNTIL = "16-NOV-2008 11:00:00" -
        /TIME = 15
$ RDB_STATS_DATABASE MYDB.RDB 2008-11-16-00-56.STATS
```

This program can be used to capture either "static" data (from a perviously collected binary file) or "real time" data where records are written to the database as they are produced from RMU /SHOW STATISTICS, as in the following example (note that the RDB_STATS_DATABASE example program should be modified when used in this fashion to commit after every record):

```
$ RDB_STATS_DATABASE := $SQL$SAMPLE:RDB_STATS_DATABASE72
$ CREATE /MAILBOX RDB_STATS_DATABASE$MAILBOX -
    /PERMANENT -
    /LOG -
    /BUFFER_SIZE = 65535 -
    /MESSAGE_SIZE = 10000
$ SPAWN RMU /SHOW STATISTICS MFP -
        /NOINTERACTIVE -
        /OUTPUT = RDB_STATS_DATABASE$MAILBOX:
        /UNTIL = "16-NOV-2009 11:00:00" -
        /TIME = 60
$ RDB_STATS_DATABASE MYDB.RDB RDB_STATS_DATABASE$MAILBOX:
```

This example is intended solely to be used as a template for writing your own program. No support for this example template program is expressed or implied.

Oracle Corporation assumes no responsibility for the functionality, correctness or use of this example program. Oracle Corporation reserves the right to change the format and contents of the Oracle Rdb RMU SHOW STATISTICS binary output file at any time without prior notice.

The RDB_STATS_DATABASE example program is comprised of the following source modules found in SQL$SAMPLE:

- RDB_STATS_DATABASExx.C
- RDB_STATS_DATABASE_SQL1_xx.SQLMOD
- RDB_STATS_DATABASE_SQL2_xx.SQLMOD

Compile and link the RDB_STATS_DATABASE example program as follows:

```
$ CC /FLOAT=IEEE RDB_STATS_DATABASExx.C
$ SQL$MOD /FLOAT=IEEE RDB_STATS_DATABASE_SQL1_xx.SQLMOD
$ SQL$MOD /FLOAT=IEEE RDB_STATS_DATABASE_SQL2_xx.SQLMOD
$ LINK RDB_STATS_DATABASExx+−
    RDB_STATS_DATABASE_SQL1_xx+−
    RDB_STATS_DATABASE_SQL2_xx+−
    SQL$USER /LIBRARY
```

5.1.13 RCS Time−Based Cache Sweeping

Previously, the Record Cache Server (RCS) process would perform modified row cache "sweep" operations only when a cache was full (also known as "clogged") with modified rows. Now a database may be configured to perform timed cache sweeps. This feature is intended to help perform "lazy" updates of modified rows to the database from caches without performing a full cache checkpoint operation.

The timer for the periodic cache sweeps is specified with the "SWEEP INTERVAL is numeric−literal seconds" clause of the ALTER DATABASE ... ROW CACHE IS ENABLED statement, as in the following example:

```
ALTER DATABASE FILENAME MF_PERSONNEL
  ROW CACHE IS ENABLED (SWEEP INTERVAL IS 300 SECONDS);
```

The number of slots per cache to sweep is specified with the ALTER CACHE statement. Legal values for "SWEEP INTERVAL" are from 0 seconds (to disable periodic timed sweeps) to 3600 seconds (1 hour).

The RMU /SET ROW_CACHE command accepts a /[NO]SWEEP_INTERVAL=n qualifier as an alternate method to specify the periodic cache sweep timer. /NOSWEEP_INTERVAL disables periodic timed sweeps and /SWEEP_INTERVAL=n can be used to set the timer for the periodic cache sweeps. Legal values for /SWEEP_INTERVAL=n are from 1 second to 3600 seconds (1 hour).

The Record Cache Server (RCS) process log file contains information about periodic row cache "sweep" operations and can be a useful analysis tool.

Default Value

The intended default value for the SWEEP INTERVAL in a database is zero seconds (meaning disabled). It is, however, possible for a database that had originally been created with Oracle Rdb Release 7.0 to have a non−zero value. Customers using the row cache feature are advised to explicitly set the SWEEP INTERVAL parameter to either zero (to disable periodic timed sweeps) or the desired sweep interval period on all databases after upgrading to Release 7.2.4.

Use RMU /SET ROW_CACHE /NOSWEEP_INTERVAL

In Oracle Rdb Release 7.2.4, the "SWEEP INTERVAL is 0 seconds" clause of the ALTER DATABASE ... ROW CACHE IS ENABLED statement may not disable the periodic row cache "sweep" operations. Oracle recommends using the RMU /SET ROW_CACHE /NOSWEEP_INTERVAL command as an alternative. This problem will be corrected in a future release.

Incomplete Display Support

Oracle Rdb Release 7.2.4 does not include complete support for showing a database's periodic row cache "sweep" operation timer value. As a workaround prior to the next Oracle Rdb release, the RMU/DUMP/HEADER/OPTIONS=DEBUG command can be used to display the database parameter RCS_SWEEP_INTERVAL as in the following example:

```
$ RMU/DUMP/HEADER/OPTIONS=DEBUG/OUTPUT=X.X MF_PERSONNEL
$ SEARCH X.X RCS_SWEEP_INTERVAL
RCACHE_CNT = 11.          RCACHE_VBN = 153.          RCS_SWEEP_INTERVAL = 123.
```

5.1.14 RMU Command TSN Keyword and Qualifier Value

RMU commands that accept a TSN keyword or qualifier value now accept input formats as follows:

- A decimal string representing a quadword TSN value
- A hexadecimal string starting with "%X" representing a quadword TSN value
- A two−part decimal string separated by a colon representing a quadword TSN value as high and low longwords

Following are some example uses of input TSN values:

```
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=54321
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=123456234253245
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=%X7655
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=%X000000715F856AB
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=0:871251
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=3:53487
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=21:653156
```

5.1.15 New Support for RENAME and CREATE SYNONYM Commands

With this release of Oracle Rdb, the RENAME and CREATE SYNONYM commands support INDEX and STORAGE MAP database objects.

- RENAME INDEX changes the name of the index in all system tables.
  A synonym is created using the old index name to reference the new name of the index. This synonym will be used by any query outline that previously referenced the index using the old name. Note that only a single synonym name may exist. Therefore, if you have indices with the same name as another object, then the RENAME INDEX command may fail if creating the synonym detects a duplicate name,
  The command ALTER INDEX ... RENAME TO ... is synonymous with the RENAME INDEX command.
- RENAME STORAGE MAP changes the name of the storage map in all system tables.

If the storage map has a companion function in the RDB$STORAGE_MAPS system module, then that function will also be renamed. A synonym is created using the old function name to reference the new name of the function. This synonym will be used by any other routine, computed by column, automatic column, and so on that referenced the old storage mapping function.
The command ALTER STORAGE MAP ... RENAME TO ... is synonymous with the RENAME STORAGE MAP command.

- CREATE SYNONYM ... FOR INDEX ... is now supported. Synonyms for indices can be created, altered and dropped.
- CREATE SYNONYM ... FOR STORAGE MAP ... is now supported. Synonyms for storage maps can be created, altered and dropped.

The following example shows the result of the RENAME INDEX and RENAME STORAGE MAP commands.

```
SQL> show table (storage maps,index) employees
Information for table EMPLOYEES

Indexes on table EMPLOYEES:
EMPLOYEES_HASH                  with column EMPLOYEE_ID
  No Duplicates allowed
  Type is Hashed Scattered
  Key suffix compression is DISABLED

EMP_EMPLOYEE_ID                 with column EMPLOYEE_ID
  No Duplicates allowed
  Type is Sorted
  Key suffix compression is DISABLED
  Node size  430

EMP_LAST_NAME                   with column LAST_NAME
  Duplicates are allowed
  Type is Sorted
  Key suffix compression is DISABLED

Storage Map for table EMPLOYEES:
     EMPLOYEES_MAP

SQL> rename storage map EMPLOYEES_MAP to EMP_STORAGE_MAP;
SQL> rename index EMPLOYEES_HASH to EMP_ID_HASH;
SQL> show table (storage maps,index) employees
Information for table EMPLOYEES


Indexes on table EMPLOYEES:
EMP_EMPLOYEE_ID                 with column EMPLOYEE_ID
  No Duplicates allowed
  Type is Sorted
  Key suffix compression is DISABLED
  Node size  430

EMP_ID_HASH                     with column EMPLOYEE_ID
  No Duplicates allowed
  Type is Hashed Scattered
  Key suffix compression is DISABLED

EMP_LAST_NAME                   with column LAST_NAME
  Duplicates are allowed
  Type is Sorted
  Key suffix compression is DISABLED
```

```
Storage Map for table EMPLOYEES:
     EMP_STORAGE_MAP

SQL> show storage map
User Storage Maps in database with filename mf_personnel_sql
     CANDIDATES_MAP
     COLLEGES_MAP
     DEGREES_MAP
     DEPARTMENTS_MAP
     EMP_STORAGE_MAP
     JOBS_MAP
     JOB_HISTORY_MAP
     SALARY_HISTORY_MAP
     WORK_STATUS_MAP
SQL> show index
User indexes in database with filename mf_personnel_sql
     COLL_COLLEGE_CODE
     DEG_COLLEGE_CODE
     DEG_EMP_ID
     DEPARTMENTS_INDEX
     EMP_EMPLOYEE_ID
     EMP_ID_HASH
     EMP_LAST_NAME
     JH_EMPLOYEE_ID
     JOB_HISTORY_HASH
     SH_EMPLOYEE_ID
     EMPLOYEES_HASH                  A synonym for index EMP_ID_HASH
SQL> show system function
Functions in database with filename mf_personnel_sql
     CANDIDATES_MAP
     COLLEGES_MAP
     DEGREES_MAP
     DEPARTMENTS_MAP
     EMP_STORAGE_MAP
     JOBS_MAP
     JOB_HISTORY_MAP
     SALARY_HISTORY_MAP
     WORK_STATUS_MAP
     EMPLOYEES_MAP                   A synonym for function EMP_STORAGE_MAP
SQL>
```

Chapter 6
Documentation Corrections, Additions and Changes

This chapter provides corrections for documentation errors and omissions.

6.1 Documentation Corrections

6.1.1 RMU /SET ROW_CACHE Command Updates

The documentation and online help for the "RMU /SET ROW_CACHE" command inadvertantly did not include the full set of allowed keywords and qualifiers.

The valid command line qualifiers for the "RMU /SET ROW_CACHE" command are:

- Alter − Specifies the action to take on the named cache. You must specify the cache name and at least one other option.
- Disable − Disables row caching. Do not use with the Enable qualifier.
- Enable − Enables row caching. Do not use with the Disable qualifier.
- Log − Specifies whether the processing of the command is reported to SYS$OUTPUT. Specify the Log qualifier to request log output and the Nolog qualifier to prevent it. If you specify neither, the default is the current setting of the DCL verify switch.
- Backing_Store_Location=devdir − Specify the per−database default backing store location.
- NoBacking_Store_Location − Remove the per−database default backing store location and revert back to the default backing store file location of the root file device and directory.

The valid values for the ALTER qualifier are:

- NAME=cachename − Name of the cache to be modified. The cache must already be defined in the database. This is a required parameter. This parameter accepts the wildcard characters asterisk (*) and percent sign (%).
- ENABLE − Enable the cache.
- DISABLE − Disable the cache.
- DROP − Drop (delete) the cache.
- SNAPSHOT_SLOT_COUNT=n − Specify the number of snaphot slots in the cache. A value of zero disables the snapshot portion for the specified cache.
- SLOT_COUNT=n − Specify the number of slots in the cache.
- SLOT_SIZE=n − Specify the size (in bytes) of each slot in the cache.
- WORKING_SET_COUNT=n − Specify the number of working set entries for the cache. Valid values are from 1 to 100.
- BACKING_STORE_LOCATION=devdir − Specify the per−cache default backing store location.
- NOBACKING_STORE_LOCATION − Remove the per−cache default backing store location and revert back to the database default backing store file location.
- SHARED_MEMORY − Specify the shared memory type and parameters for the cache. Valid keywords are:
    - ♦ TYPE=PROCESS to specify traditional shared memory global section, which means that the database global section is located in process (P0) address space and may be paged from the processes working set as needed.
    - ♦ TYPE=RESIDENT to specify that the database global section is memory resident in process (P0) address space using OpenVMS Alpha shared page tables, which means that a system space global section is fully resident, or pinned, in memory.
    - ♦ RAD_HINT= "number" to indicate a request that memory for this shared memory should be allocated from the specified OpenVMS Alpha Resource Affinity Domain (RAD). This parameter specifies a hint to Oracle Rdb and OpenVMS about where memory should be

physically allocated. It is possible that if the memory is not available, it will be allocated from other RADs in the system. For systems that do not support RADs, no RAD_HINT specification is valid.
The RAD_HINT qualifier is only valid when the shared memory type is set to RESIDENT. Setting the shared memory type to SYSTEM or PROCESS explicitly disables any previously defined RAD hint.

Note

OpenVMS support for RADs is available only on the AlphaServer GS series systems. For more information about using RADs, refer to the OpenVMS Alpha Partitioning and Galaxy Guide.

♦ NORAD_HINT disables the RAD hint.

The "/ALTER=(...)" qualifier may be specified multiple times on the command line. Each /ALTER qualifier specified operates on one unique cache if no wildcard character (% or *) is specified. Otherwise, Each /ALTER operates on all matching cache names.

For example, the following command alters two caches:

```
$ RMU /SET ROW_CACHE MF_PERSONNEL -
        /ALTER= (   NAME = RDB$SYS_CACHE,
                    SLOT_COUNT = 800) -
        /ALTER= (   NAME = RESUMES, -
                    SLOT_SIZE=500, -
                    WORKING_SET_COUNT = 15)
```

The following command alters caches named FOOD and FOOT (and any other cache with a 4 character name with the first three characters of "FOO" defined in the database):

```
$ RMU /SET ROW_CACHE MF_PERSONNEL -
        /ALTER= (   NAME = FOO%,
                    BACKING_STORE_LOCATION=DISK$RDC:[RDC])
```

6.1.2 Documentation for the DEBUG_OPTIONS Qualifier of RMU Unload

Bug 8447357

The RMU Help file and RMU Reference Manual is missing the description of the following qualifier for RMU Unload.

The DEBUG_OPTIONS qualifier accepts a list of keyword options.

- [NO]TRACE
  Traces the qualifier and parameter processing performed by RMU Unload. In addition, the query executed to read the table data is annotated with the TRACE statement at each Commit (controlled by Commit_Every qualifier). When the logical name RDMS$SET_FLAGS is defined as "TRACE", then a line similar to the following is output after each commit is performed.

```
~Xt: 2009−04−23 15:16:16.95: Commit executed.
```

The default is NOTRACE.

```
$ RMU/UNLOAD/REC=(FILE=WS,FORMAT=CONTROL) SQL$DATABASE WORK_STATUS WS/DEBUG=
TRACE
Debug = TRACE
* Synonyms are not enabled
Row_Count = 500
Message buffer: Len: 13524
Message buffer: Sze: 27, Cnt: 500, Use: 4 Flg: 00000000
%RMU−I−DATRECUNL,   3 data records unloaded.
```

- [NO]FILENAME_ONLY
  When the qualifier Record_Definition=Format:CONTROL is used, the name of the created unload file is written to the control file (.CTL). When the keyword FILENAME_ONLY is specified, RMU Unload will prune the output file specification to show only the file name and type. The default is NOFILENAME_ONLY.

```
$ RMU/UNLOAD/REC=(FILE=TT:,FORMAT=CONTROL) SQL$DATABASE WORK_STATUS WS/DEBUG=
FILENAME
--
-- SQL*Loader Control File
--   Generated by: RMU/UNLOAD
--   Version:      Oracle Rdb X7.2−00
--   On:           23−APR−2009 11:12:46.29
--
LOAD DATA
INFILE 'WS.UNL'
APPEND
INTO TABLE "WORK_STATUS"
(
 STATUS_CODE                     POSITION(1:1) CHAR NULLIF (RDB$UL_NB1 = '1')
,STATUS_NAME                     POSITION(2:9) CHAR NULLIF (RDB$UL_NB2 = '1')
,STATUS_TYPE                     POSITION(10:23) CHAR NULLIF (RDB$UL_NB3 = '1')
-- NULL indicators
,RDB$UL_NB1           FILLER POSITION(24:24) CHAR -- indicator for STATUS_CODE
,RDB$UL_NB2           FILLER POSITION(25:25) CHAR -- indicator for STATUS_NAME
,RDB$UL_NB3           FILLER POSITION(26:26) CHAR -- indicator for STATUS_TYPE
)
%RMU−I−DATRECUNL,   3 data records unloaded.
```

- [NO]HEADER
  This keyword controls the output of the header in the control file. To suppress the header, use NOHEADER. The default is HEADER.
- APPEND, INSERT, REPLACE, TRUNCATE
  These keywords control the text that is output prior to the INTO TABLE clause in the control file. The default is APPEND and only one of these options can be specified.

6.1.3 SQL$MSGxx.DOC Is Not Alphabetical

Bug 4387383

The last paragraph of page A−3 of volume 5 of the SQL Reference Manual says the message codes in files such as SQL$MSG71.DOC are alphabetized. However, it was found that the message codes were not alphabetized in SQL$MSG71.DOC or SQL$MSG72.DOC.

The cause of this problem was that the COSI message codes were appended to the end of the SQL message codes in this file.

This has been corrected in Oracle Rdb Release 7.2.4. We no longer append the COSI message codes to the SQL$MSGnn.DOC file since the COSI message codes are available separately.

6.1.4 LOCK_TIMEOUT Documentation Error in RMU Reference Manual Release 7.2

The "Oracle Rdb for OpenVMS: Oracle RMU Reference Manual Release 7.2" incorrectly implies that there is a default value for the lock timeout in seconds specified by the /LOCK_TIMEOUT qualifier in the following sections:

- 1.10 RMU/BACKUP COMMAND
- 1.11 RMU/BACKUP/AFTER_JOURNAL COMMAND
- 1.17 RMU COPY_DATABASE COMMAND

In all these sections, in the description of the "/Lock_Timeout=n" qualifier, any reference to a default value such as "The default value for the /Lock_Timeout=n qualifier is ..." needs to be removed since there is no default value allowed for this qualifier. If you specifiy the /LOCK_TIMEOUT qualifier, you have to specify the lock timeout value in seconds. If you do not specifiy the /LOCK_TIMEOUT qualifier, the default is to wait indefinitely to acquire the QUIET POINT lock and any other locks needed for ONLINE execution of the command. It should also be mentioned that the LOCK_TIMEOUT value does not only affect the QUIET POINT lock but can affect other locks RMU may need to acquire for ONLINE execution.

6.1.5 Revised Example for SET OPTIMIZATION LEVEL Statement

Bug 6350960

Example 1: Setting the optimization level

The dynamic optimizer can use either FAST FIRST or TOTAL TIME tactics to return rows to the application. The default setting, FAST FIRST, assumes that applications, especially those using interactive SQL, will want to see rows as quickly as possible and possibly abort the query before completion. Therefore, if the FAST FIRST tactic is possible, the optimizer will sacrifice overall retrieval time to initially return rows quickly. This choice can be affected by setting the OPTIMIZATION LEVEL.

The following example contrasts the query strategies selected when FAST FIRST versus TOTAL TIME is in effect. Databases and queries will vary in their requirements. Queries should be tuned to see which setting best suits the needs of the application environment. For the MF_PERSONNEL database, there is little or no difference between these tactics but for larger tables the differences could be noticeable.

```
SQL> set flags 'STRATEGY,DETAIL';
SQL> --
SQL> -- No optimization level has been selected. The optimizer
SQL> -- selects the FAST FIRST (FFirst) retrieval tactic to
SQL> -- retrieve the rows from the EMPLOYEES table in the
SQL> -- following query:
SQL> --
SQL> select EMPLOYEE_ID, LAST_NAME
cont> from EMPLOYEES
cont> where EMPLOYEE_ID IN ('00167', '00168');
Tables:
  0 = EMPLOYEES
Leaf#01 FFirst 0:EMPLOYEES Card=100
  Bool: (0.EMPLOYEE_ID = '00167') OR (0.EMPLOYEE_ID = '00168')
  BgrNdx1 EMPLOYEES_HASH [(1:1)2] Fan=1
    Keys: r0: 0.EMPLOYEE_ID = '00168'
          r1: 0.EMPLOYEE_ID = '00167'
 EMPLOYEE_ID   LAST_NAME
 00167         Kilpatrick
 00168         Nash
2 rows selected
SQL> --
SQL> -- Use the SET OPTIMIZATION LEVEL statement to specify that
SQL> -- you want the TOTAL TIME (BgrOnly) retrieval strategy to
SQL> -- be used.
SQL> --
SQL> SET OPTIMIZATION LEVEL 'TOTAL TIME';
SQL> select EMPLOYEE_ID, LAST_NAME
cont> from EMPLOYEES
cont> where EMPLOYEE_ID IN ('00167', '00168');
Tables:
  0 = EMPLOYEES
Leaf#01 BgrOnly 0:EMPLOYEES Card=100
  Bool: (0.EMPLOYEE_ID = '00167') OR (0.EMPLOYEE_ID = '00168')
  BgrNdx1 EMPLOYEES_HASH [(1:1)2] Fan=1
    Keys: r0: 0.EMPLOYEE_ID = '00168'
          r1: 0.EMPLOYEE_ID = '00167'
 EMPLOYEE_ID   LAST_NAME
 00167         Kilpatrick
 00168         Nash
2 rows selected
SQL> --
SQL> -- When the SET OPTIMIZATION LEVEL 'DEFAULT' statement
SQL> -- is specified the session will revert to the default FAST FIRST
SQL> -- optimizer tactic.
SQL> --
SQL> SET OPTIMIZATION LEVEL 'DEFAULT';
SQL> select EMPLOYEE_ID, LAST_NAME
cont> from EMPLOYEES
cont> where EMPLOYEE_ID IN ('00167', '00168');
Tables:
  0 = EMPLOYEES
Leaf#01 FFirst 0:EMPLOYEES Card=100
  Bool: (0.EMPLOYEE_ID = '00167') OR (0.EMPLOYEE_ID = '00168')
  BgrNdx1 EMPLOYEES_HASH [(1:1)2] Fan=1
    Keys: r0: 0.EMPLOYEE_ID = '00168'
          r1: 0.EMPLOYEE_ID = '00167'
 EMPLOYEE_ID   LAST_NAME
 00167         Kilpatrick
 00168         Nash
2 rows selected
SQL>
```

6.1.6 RMU /VERIFY Process Quotas and Limits Clarification

When using the RMU/VERIFY command, a process requires a minimum of the following quotas:

- FILLM and CHANNELCNT at least 25 more than the total number of database storage areas, snapshot storage areas, and after image journals.
- Large enough BYTLM, page file quota and working set to open all of the database storage areas, snapshot storage areas, and after image journals.

6.1.7 Online Backup Can Be Performed With Transfer Via Memory

The following incorrect Oracle RMU BACKUP command restriction will be removed from the Oracle RMU Reference Manual.

In prior releases of the Oracle RMU Reference Manual, it states under the RMU Backup Online option that "However, an online backup operation cannot be performed if TRANSFER VIA MEMORY, also referred to as optimized page transfer, is enabled. (See the description of the SQL ALTER DATABASE statement in the Oracle Rdb SQL Reference Manual for information on optimized page transfer.)". This restriction is no longer true and will be removed from the Oracle RMU Reference Manual.

The same restriction is also listed for the Online Copy Database and for the Online Move Area commands. This restriction is no longer in place for these commands so it will be removed from the Oracle RMU Reference Manual.

6.1.8 Missing Example for CREATE STORAGE MAP

Bug 5655348

The SQL Reference Manual did not include an example showing the storage area attributes for a LIST storage map. The following example will appear in a future version of the Oracle Rdb V7.2 SQL Reference Manual in the CREATE STORAGE MAP section.

Example

The following example shows the use of storage area attributes in a LIST storage map. The storage area attributes must be immediately following the storage area name (as in table storage maps).

```
SQL> create database
cont>     filename 'DB$:MULTIMEDIA'
cont>
cont>     create storage area PHOTO_AREA1
cont>         filename 'DB$:PHOTO_AREA1'
cont>         page format UNIFORM
cont>
cont>     create storage area PHOTO_AREA2
cont>         filename 'DB$:PHOTO_AREA2'
cont>         page format UNIFORM
cont>
cont>     create storage area TEXT_AREA
cont>         filename 'DB$:TEXT_AREA'
cont>         page format UNIFORM
cont>
cont>     create storage area AUDIO_AREA
cont>         filename 'DB$:AUDIO_AREA'
cont>         page format UNIFORM
cont>
cont>     create storage area DATA_AREA
cont>         filename 'DB$:DATA_AREA'
cont>         page format UNIFORM
cont> ;
SQL>
SQL> create table EMPLOYEES
cont>     (name       char(30),
cont>      dob        date,
cont>      ident      integer,
cont>      photograph list of byte varying (4096) as binary,
cont>      resume     list of byte varying (132) as text,
cont>      review     list of byte varying (80) as text,
cont>      voiceprint list of byte varying (4096) as binary
cont>     );
SQL>
SQL> create storage map EMPLOYEES_MAP
cont>     for EMPLOYEES
cont>     enable compression
cont>     store in DATA_AREA;f
SQL>
SQL> create storage map LISTS_MAP
cont>     store lists
cont>         in AUDIO_AREA
cont>                 (thresholds are (89, 99, 100)
cont>                 ,comment is 'The voice clips'
cont>                 ,partition AUDIO_STUFF)
cont>             for (employees.voiceprint)
cont>         in TEXT_AREA
cont>                 (thresholds is (99)
cont>                 ,partition TEXT_DOCUMENTS)
cont>             for (employees.resume, employees.review)
cont>         in (PHOTO_AREA1
cont>                 (comment is 'Happy Smiling Faces?'
cont>                 ,threshold is (99)
cont>                 ,partition PHOTOGRAPHIC_IMAGES_1)
cont>             ,PHOTO_AREA2
cont>                 (comment is 'Happy Smiling Faces?'
cont>                 ,threshold is (99)
cont>                 ,partition PHOTOGRAPHIC_IMAGES_2)
cont>             )
cont>             for (employees.photograph)
cont>             fill randomly
cont>         in RDB$SYSTEM
cont>                 (partition SYSTEM_LARGE_OBJECTS);
SQL>
SQL> show storage map LISTS_MAP;
     LISTS_MAP
 For Lists
 Store clause:          STORE lists
        in AUDIO_AREA
                (thresholds are (89, 99, 100)
                ,comment is 'The voice clips'
                ,partition AUDIO_STUFF)
            for (employees.voiceprint)
        in TEXT_AREA
                (thresholds is (99)
                ,partition TEXT_DOCUMENTS)
            for (employees.resume, employees.review)
        in (PHOTO_AREA1
                (comment is 'Happy Smiling Faces?'
                ,threshold is (99)
                ,partition PHOTOGRAPHIC_IMAGES_1)
            ,PHOTO_AREA2
                (comment is 'Happy Smiling Faces?'
                ,threshold is (99)
                ,partition PHOTOGRAPHIC_IMAGES_2)
            )
            for (employees.photograph)
            fill randomly
        in RDB$SYSTEM
                (partition SYSTEM_LARGE_OBJECTS)

 Partition information for lists map:
 Vertical Partition: VRP_P000
  Partition: (1) AUDIO_STUFF
    Fill Randomly
   Storage Area: AUDIO_AREA
         Thresholds are (89, 99, 100)
 Comment:       The voice clips
  Partition: (2) TEXT_DOCUMENTS
    Fill Randomly
   Storage Area: TEXT_AREA
         Thresholds are (99, 100, 100)
  Partition: (3) PHOTOGRAPHIC_IMAGES_1
    Fill Randomly
   Storage Area: PHOTO_AREA1
         Thresholds are (99, 100, 100)
 Comment:       Happy Smiling Faces?
  Partition: (3) PHOTOGRAPHIC_IMAGES_2
   Storage Area: PHOTO_AREA2
         Thresholds are (99, 100, 100)
 Comment:       Happy Smiling Faces?
  Partition: (4) SYSTEM_LARGE_OBJECTS
    Fill Randomly
   Storage Area: RDB$SYSTEM
SQL>
SQL> commit;
```

6.1.9 RDM$BIND_MAX_DBR_COUNT Documentation Clarification

Bugs 1495227 and 3916606

The Rdb7 Guide to Database Performance and Tuning Manual, Volume 2, page A−18, incorrectly describes the use of the RDM$BIND_MAX_DBR_COUNT logical.

Following is an updated description. Note that the difference in actual behavior between what is in the existing documentation and software is that the logical name only controls the number of database recovery processes created at once during "node failure" recovery (that is, after a system or monitor crash or other abnormal shutdown) for each database.

When an entire database is abnormally shut down (due, for example, to a system failure), the database will have to be recovered in a "node failure" recovery mode. This recovery will be performed by another monitor in the cluster if the database is opened on another node or will be performed the next time the database is opened.

The RDM$BIND_MAX_DBR_COUNT logical name and the RDB_BIND_MAX_DBR_COUNT configuration parameter define the maximum number of database recovery (DBR) processes to be simultaneously invoked by the database monitor for each database during a "node failure" recovery.

This logical name and configuration parameter apply only to databases that do not have global buffers enabled. Databases that utilize global buffers have only one recovery process started at a time during a "node failure" recovery.

In a node failure recovery situation with the Row Cache feature enabled (regardless of the global buffer state), the database monitor will start a single database recovery (DBR) process to recover the Row Cache Server (RCS) process and all user processes from the oldest active checkpoint in the database.

Per−Database Value

The RDM$BIND_MAX_DBR_COUNT logical name specifies the maximum number of database recovery processes to run at once for each database. For example, if there are 10 databases being recovered and the value for the RDM$BIND_MAX_DBR_COUNT logical name is 8, up to 80 database recovery processes would be started by the monitor after a node failure.

The RDM$BIND_MAX_DBR_COUNT logical name is translated when the monitor process opens a database. Databases need to be closed and reopened for a new value of the logical to become effective.

6.1.10 Database Server Process Priority Clarification

By default, the database servers (ABS, ALS, DBR, LCS, LRS, RCS) created by the Rdb monitor inherit their VMS process scheduling base priority from the Rdb monitor process. The default priority for the Rdb monitor process is 15.

Individual server priorities can be explicitly controlled via system−wide logical names as described in Table 6−1.

Table 6−1 Server Process Priority Logical Names

Logical Name	Use
RDM$BIND_ABS_PRIORITY	Base Priority for the ABS Server process
RDM$BIND_ALS_PRIORITY	Base Priority for the ALS Server process
RDM$BIND_DBR_PRIORITY	Base Priority for the DBR Server process
RDM$BIND_LCS_PRIORITY	Base Priority for the LCS Server process
RDM$BIND_LRS_PRIORITY	Base Priority for the LRS Server process
RDM$BIND_RCS_PRIORITY	Base Priority for the RCS Server process

When the Hot Standby feature is installed, the RDMAIJSERVER account is created specifying an account priority of 15. The priority of AIJ server processes on your system can be restricted with the system−wide logical name RDM$BIND_AIJSRV_PRIORITY. If this logical name is defined to a value less than 15, an AIJ server process will adjust its base priority to the value specified when the AIJ server process starts. Values from 0 to 31 are allowed for RDM$BIND_AIJSRV_PRIORITY, but the process is not able to raise its priority above the RDMAIJSERVER account value.

For most applications and systems, Oracle discourages changing the server process priorities.

6.1.11 Explanation of SQL$INT in a SQL Multiversion Environment and How to Redefine SQL$INT

Bug 2500594

In an environment running multiple versions of Oracle Rdb, for instance Rdb V7.0 and Rdb V7.1, there are now several varianted SQL images, such as SQL$70.EXE and SQL$71.EXE. However, SQL$INT.EXE is not varianted but acts as a dispatcher using the translation of the logical name RDMS$VERSION_VARIANT to activate the correct SQL runtime environment. This image is replaced when a higher version of Oracle Rdb is installed. Thus, using the example above, when Rdb V7.1 is installed, SQL$INT.EXE will be replaced with the V7.1 SQL$INT.EXE.

If an application is linked in this environment (using V7.1 SQL$INT) and the corresponding executable deployed to a system running Oracle Rdb V7.0 multiversion only, the execution of the application may result in the following error:

```
%IMGACT−F−SYMVECMIS, shareable image's symbol vector table mismatch
```

In order to avoid such a problem, the following alternative is suggested:

In the multiversion environment running both Oracle Rdb V7.0 and Oracle Rdb V7.1, run Oracle Rdb V7.0 multiversion by running the command procedures RDB$SETVER.COM 70 and RDB$SETVER RESET. This will set up the necessary logical names and symbols that establish the Oracle Rdb V7.0 environment.

For example:

```
$ @SYS$LIBRARY:RDB$SETVER 70

Current PROCESS Oracle Rdb environment is version V7.0−63 (MULTIVERSION)
Current PROCESS SQL environment is version V7.0−63 (MULTIVERSION)
Current PROCESS Rdb/Dispatch environment is version V7.0−63 (MULTIVERSION)

$ @SYS$LIBRARY:RDB$SERVER RESET
```

Now run SQL and verify that the version is correct:

```
$ sql$
SQL> show version
Current version of SQL is: Oracle Rdb SQL V7.0−63
```

Define SQL$INT to point to the varianted SQL$SHR.EXE. Then, create an options file directing the linker to link with this newly defined SQL$INT. An example follows:

```
$ DEFINE SQL$INT SYS$SHARE:SQL$SHR'RDMS$VERSION_VARIANT'.EXE
$ LINK TEST_APPL,SQL$USER/LIB,SYS$INPUT/option
SQL$INT/SHARE
```

```
^Z
```

The executable is now ready to be deployed to the Oracle Rdb V7.0 multiversion environment and should run successfully.

Please note that with each release of Oracle Rdb, new entry points are added to the SQL$INT shareable image. This allows the implementation of new functionality. Therefore, applications linked with SQL$INT from Oracle Rdb V7.1 cannot be run on systems with only Oracle Rdb V7.0 installed. This is because the shareable image does not contain sufficient entry points.

The workaround presented here allows an application to explicitly link with the Oracle Rdb V7.0 version of the image. Such applications are upward compatible and will run on Oracle Rdb V7.0 and Oracle Rdb V7.1. The applications should be compiled and linked under the lowest version.

In environments where Oracle Rdb V7.1 is installed, this workaround is not required because the SQL$INT image will dynamically activate the appropriate SQL$SHRxx image as expected.

6.1.12 Clarification of PREPARE Statement Behavior

Bug 2581863

According to the Oracle Rdb7 SQL Reference Manual, Volume 3 page 7−227, when using a statement−id parameter for PREPARE "if that parameter is an integer, then you must explicitly initialize that integer to zero before executing the PREPARE statement".

This description is not correct and should be replaced with this information:

1. If the statement−id is non−zero and does not match any prepared statement (the id was stale or contained a random value), then an error is raised:
   %SQL−F−BADPREPARE, Cannot use DESCRIBE or EXECUTE on a statement that is not prepared
2. If the statement−id is non−zero, or the statement name is one that has previously been used and matches an existing prepared statement, then that statement is automatically released prior to the prepare of the new statement. Please refer to the RELEASE statement for further details.
3. If the statement−id is zero or was automatically released, then a new statement−id is allocated and the statement prepared.

Please note that if you use statement−name instead of a statement−id−parameter then SQL will implicitly declare an id for use by the application. Therefore, the semantics described apply similarly when using the statement−name. See the RELEASE statement for details.

6.1.13 RDM$BIND_LOCK_TIMEOUT_INTERVAL Overrides the Database Parameter

Bug 2203700

When starting a transaction, there are three different values that are used to determine the lock timeout interval for that transaction. Those values are:

1. The value specified in the SET TRANSACTION statement

2. The value stored in the database as specified in CREATE or ALTER DATABASE
3. The value of the logical name RDM$BIND_LOCK_TIMEOUT_INTERVAL

The timeout interval for a transaction is the smaller of the value specified in the SET TRANSACTION statement and the value specified in CREATE DATABASE. However, if the logical name RDM$BIND_LOCK_TIMEOUT_INTERVAL is defined, the value of this logical name overrides the value specified in CREATE DATABASE.

The description of how these three values interact, found in several different parts of the Rdb documentation set, is incorrect and will be replaced by the description above.

The lock timeout value in the database can be dynamically modified from the Locking Dashboard in RMU/SHOW STATISTICS. The Per−Process Locking Dashboard can be used to dynamically override the logical name RDM$BIND_LOCK_TIMEOUT_INTERVAL for one or more processes.

6.1.14 Missing Tables Descriptions for the RDBEXPERT Collection Class

Appendix B in the Oracle Rdb7 Guide to Database Performance and Tuning describes the event−based data tables in the formatted database for the Oracle Rdb PERFORMANCE and RDBEXPERT collection classes. This section describes the missing tables for the RDBEXPERT collection class.

Table 6−2 shows the TRANS_TPB table.

Table 6−2 Columns for Table EPC$1_221_TRANS_TPB

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_POINT	DATE VMS	
CLIENT_PC	INTEGER	
STREAM_ID	INTEGER	
TRANS_ID	VARCHAR(16)	
TRANS_ID_STR_ID	INTEGER	STR_ID_DOMAIN
TPB	VARCHAR(127)	
TPB_STR_ID	INTEGER	STR_ID_DOMAIN

Table 6−3 shows the TRANS_TPB_ST table. An index is provided for this table. It is defined with column STR_ID, duplicates are allowed, and the type is sorted.

Table 6−3 Columns for Table EPC$1_221_TRANS_TPB_ST

Column Name	Data Type	Domain
STR_ID	INTEGER	STR_ID_DOMAIN

SEGMENT_NUMBER	SMALLINT	SEGMENT_NUMBER_DOMAIN
STR_SEGMENT	VARCHAR(128)	

6.1.15 Missing Columns Descriptions for Tables in the Formatted Database

Some of the columns were missing from the tables in Appendix B in the Oracle Rdb7 Guide to Database Performance and Tuning. The complete table definitions are described in this section.

Table 6−4 shows the DATABASE table.

Table 6−4 Columns for Table EPC$1_221_DATABASE

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_POINT	DATE VMS	
CLIENT_PC	INTEGER	
STREAM_ID	INTEGER	
DB_NAME	VARCHAR(255)	
DB_NAME_STR_ID	INTEGER	STR_ID_DOMAIN
IMAGE_FILE_NAME	VARCHAR(255)	
IMAGE_FILE_NAME_STR_ID	INTEGER	STR_ID_DOMAIN

Table 6−5 shows the REQUEST_ACTUAL table.

Table 6−5 Columns for Table EPC$1_221_REQUEST_ACTUAL

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_START	DATE VMS	
TIMESTAMP_END	DATE VMS	
DBS_READS_START	INTEGER	
DBS_WRITES_START	INTEGER	
RUJ_READS_START	INTEGER	
RUJ_WRITES_START	INTEGER	
AIJ_WRITES_START	INTEGER	
ROOT_READS_START	INTEGER	

ROOT_WRITES_START	INTEGER
BUFFER_READS_START	INTEGER
GET_VM_BYTES_START	INTEGER
FREE_VM_BYTES_START	INTEGER
LOCK_REQS_START	INTEGER
REQ_NOT_QUEUED_START	INTEGER
REQ_STALLS_START	INTEGER
REQ_DEADLOCKS_START	INTEGER
PROM_DEADLOCKS_START	INTEGER
LOCK_RELS_START	INTEGER
LOCK_STALL_TIME_START	INTEGER
D_FETCH_RET_START	INTEGER
D_FETCH_UPD_START	INTEGER
D_LB_ALLOK_START	INTEGER
D_LB_GBNEEDLOCK_START	INTEGER
D_LB_NEEDLOCK_START	INTEGER
D_LB_OLDVER_START	INTEGER
D_GB_NEEDLOCK_START	INTEGER
D_GB_OLDVER_START	INTEGER
D_NOTFOUND_IO_START	INTEGER
D_NOTFOUND_SYN_START	INTEGER
S_FETCH_RET_START	INTEGER
S_FETCH_UPD_START	INTEGER
S_LB_ALLOK_START	INTEGER
S_LB_GBNEEDLOCK_START	INTEGER
S_LB_NEEDLOCK_START	INTEGER
S_LB_OLDVER_START	INTEGER
S_GB_NEEDLOCK_START	INTEGER
S_GB_OLDVER_START	INTEGER
S_NOTFOUND_IO_START	INTEGER
S_NOTFOUND_SYN_START	INTEGER
D_ASYNC_FETCH_START	INTEGER
S_ASYNC_FETCH_START	INTEGER
D_ASYNC_READIO_START	INTEGER
S_ASYNC_READIO_START	INTEGER
AS_READ_STALL_START	INTEGER
AS_BATCH_WRITE_START	INTEGER
AS_WRITE_STALL_START	INTEGER
BIO_START	INTEGER
DIO_START	INTEGER
PAGEFAULTS_START	INTEGER
PAGEFAULT_IO_START	INTEGER

CPU_START	INTEGER	
CURRENT_PRIO_START	SMALLINT	
VIRTUAL_SIZE_START	INTEGER	
WS_SIZE_START	INTEGER	
WS_PRIVATE_START	INTEGER	
WS_GLOBAL_START	INTEGER	
CLIENT_PC_END	INTEGER	
STREAM_ID_END	INTEGER	
REQ_ID_END	INTEGER	
COMP_STATUS_END	INTEGER	
REQUEST_OPER_END	INTEGER	
TRANS_ID_END	VARCHAR(16)	
TRANS_ID_END_STR_ID	INTEGER	STR_ID_DOMAIN
DBS_READS_END	INTEGER	
DBS_WRITES_END	INTEGER	
RUJ_READS_END	INTEGER	
RUJ_WRITES_END	INTEGER	
AIJ_WRITES_END	INTEGER	
ROOT_READS_END	INTEGER	
ROOT_WRITES_END	INTEGER	
BUFFER_READS_END	INTEGER	
GET_VM_BYTES_END	INTEGER	
FREE_VM_BYTES_END	INTEGER	
LOCK_REQS_END	INTEGER	
REQ_NOT_QUEUED_END	INTEGER	
REQ_STALLS_END	INTEGER	
REQ_DEADLOCKS_END	INTEGER	
PROM_DEADLOCKS_END	INTEGER	
LOCK_RELS_END	INTEGER	
LOCK_STALL_TIME_END	INTEGER	
D_FETCH_RET_END	INTEGER	
D_FETCH_UPD_END	INTEGER	
D_LB_ALLOK_END	INTEGER	
D_LB_GBNEEDLOCK_END	INTEGER	
D_LB_NEEDLOCK_END	INTEGER	
D_LB_OLDVER_END	INTEGER	
D_GB_NEEDLOCK_END	INTEGER	
D_GB_OLDVER_END	INTEGER	
D_NOTFOUND_IO_END	INTEGER	
D_NOTFOUND_SYN_END	INTEGER	
S_FETCH_RET_END	INTEGER	
S_FETCH_UPD_END	INTEGER	

S_LB_ALLOK_END	INTEGER
S_LB_GBNEEDLOCK_END	INTEGER
S_LB_NEEDLOCK_END	INTEGER
S_LB_OLDVER_END	INTEGER
S_GB_NEEDLOCK_END	INTEGER
S_GB_OLDVER_END	INTEGER
S_NOTFOUND_IO_END	INTEGER
S_NOTFOUND_SYN_END	INTEGER
D_ASYNC_FETCH_END	INTEGER
S_ASYNC_FETCH_END	INTEGER
D_ASYNC_READIO_END	INTEGER
S_ASYNC_READIO_END	INTEGER
AS_READ_STALL_END	INTEGER
AS_BATCH_WRITE_END	INTEGER
AS_WRITE_STALL_END	INTEGER
BIO_END	INTEGER
DIO_END	INTEGER
PAGEFAULTS_END	INTEGER
PAGEFAULT_IO_END	INTEGER
CPU_END	INTEGER
CURRENT_PRIO_END	SMALLINT
VIRTUAL_SIZE_END	INTEGER
WS_SIZE_END	INTEGER
WS_PRIVATE_END	INTEGER
WS_GLOBAL_END	INTEGER

Table 6−6 shows the TRANSACTION table.

Table 6−6 Columns for Table EPC$1_221_TRANSACTION

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_START	DATE VMS	
TIMESTAMP_END	DATE VMS	
CLIENT_PC_START	INTEGER	
STREAM_ID_START	INTEGER	
LOCK_MODE_START	INTEGER	
TRANS_ID_START	VARCHAR(16)	
TRANS_ID_START_STR_ID	INTEGER	STR_ID_DOMAIN
GLOBAL_TID_START	VARCHAR(16)	

GLOBAL_TID_START_STR_ID	INTEGER	STR_ID_DOMAIN
DBS_READS_START	INTEGER	
DBS_WRITES_START	INTEGER	
RUJ_READS_START	INTEGER	
RUJ_WRITES_START	INTEGER	
AIJ_WRITES_START	INTEGER	
ROOT_READS_START	INTEGER	
ROOT_WRITES_START	INTEGER	
BUFFER_READS_START	INTEGER	
GET_VM_BYTES_START	INTEGER	
FREE_VM_BYTES_START	INTEGER	
LOCK_REQS_START	INTEGER	
REQ_NOT_QUEUED_START	INTEGER	
REQ_STALLS_START	INTEGER	
REQ_DEADLOCKS_START	INTEGER	
PROM_DEADLOCKS_START	INTEGER	
LOCK_RELS_START	INTEGER	
LOCK_STALL_TIME_START	INTEGER	
D_FETCH_RET_START	INTEGER	
D_FETCH_UPD_START	INTEGER	
D_LB_ALLOK_START	INTEGER	
D_LB_GBNEEDLOCK_START	INTEGER	
D_LB_NEEDLOCK_START	INTEGER	
D_LB_OLDVER_START	INTEGER	
D_GB_NEEDLOCK_START	INTEGER	
D_GB_OLDVER_START	INTEGER	
D_NOTFOUND_IO_START	INTEGER	
D_NOTFOUND_SYN_START	INTEGER	
S_FETCH_RET_START	INTEGER	
S_FETCH_UPD_START	INTEGER	
S_LB_ALLOK_START	INTEGER	
S_LB_GBNEEDLOCK_START	INTEGER	
S_LB_NEEDLOCK_START	INTEGER	
S_LB_OLDVER_START	INTEGER	
S_GB_NEEDLOCK_START	INTEGER	
S_GB_OLDVER_START	INTEGER	
S_NOTFOUND_IO_START	INTEGER	
S_NOTFOUND_SYN_START	INTEGER	
D_ASYNC_FETCH_START	INTEGER	
S_ASYNC_FETCH_START	INTEGER	
D_ASYNC_READIO_START	INTEGER	
S_ASYNC_READIO_START	INTEGER	

AS_READ_STALL_START	INTEGER	
AS_BATCH_WRITE_START	INTEGER	
AS_WRITE_STALL_START	INTEGER	
AREA_ITEMS_START	VARCHAR(128)	
AREA_ITEMS_START_STR_ID	INTEGER	STR_ID_DOMAIN
BIO_START	INTEGER	
DIO_START	INTEGER	
PAGEFAULTS_START	INTEGER	
PAGEFAULT_IO_START	INTEGER	
CPU_START	INTEGER	
CURRENT_PRIO_START	SMALLINT	
VIRTUAL_SIZE_START	INTEGER	
WS_SIZE_START	INTEGER	
WS_PRIVATE_START	INTEGER	
WS_GLOBAL_START	INTEGER	
CROSS_FAC_2_START	INTEGER	
CROSS_FAC_3_START	INTEGER	
CROSS_FAC_7_START	INTEGER	
CROSS_FAC_14_START	INTEGER	
DBS_READS_END	INTEGER	
DBS_WRITES_END	INTEGER	
RUJ_READS_END	INTEGER	
RUJ_WRITES_END	INTEGER	
AIJ_WRITES_END	INTEGER	
ROOT_READS_END	INTEGER	
ROOT_WRITES_END	INTEGER	
BUFFER_READS_END	INTEGER	
GET_VM_BYTES_END	INTEGER	
FREE_VM_BYTES_END	INTEGER	
LOCK_REQS_END	INTEGER	
REQ_NOT_QUEUED_END	INTEGER	
REQ_STALLS_END	INTEGER	
REQ_DEADLOCKS_END	INTEGER	
PROM_DEADLOCKS_END	INTEGER	
LOCK_RELS_END	INTEGER	
LOCK_STALL_TIME_END	INTEGER	
D_FETCH_RET_END	INTEGER	
D_FETCH_UPD_END	INTEGER	
D_LB_ALLOK_END	INTEGER	
D_LB_GBNEEDLOCK_END	INTEGER	
D_LB_NEEDLOCK_END	INTEGER	
D_LB_OLDVER_END	INTEGER	

D_GB_NEEDLOCK_END	INTEGER	
D_GB_OLDVER_END	INTEGER	
D_NOTFOUND_IO_END	INTEGER	
D_NOTFOUND_SYN_END	INTEGER	
S_FETCH_RET_END	INTEGER	
S_FETCH_UPD_END	INTEGER	
S_LB_ALLOK_END	INTEGER	
S_LB_GBNEEDLOCK_END	INTEGER	
S_LB_NEEDLOCK_END	INTEGER	
S_LB_OLDVER_END	INTEGER	
S_GB_NEEDLOCK_END	INTEGER	
S_GB_OLDVER_END	INTEGER	
S_NOTFOUND_IO_END	INTEGER	
S_NOTFOUND_SYN_END	INTEGER	
D_ASYNC_FETCH_END	INTEGER	
S_ASYNC_FETCH_END	INTEGER	
D_ASYNC_READIO_END	INTEGER	
S_ASYNC_READIO_END	INTEGER	
AS_READ_STALL_END	INTEGER	
AS_BATCH_WRITE_END	INTEGER	
AS_WRITE_STALL_END	INTEGER	
AREA_ITEMS_END	VARCHAR(128)	
AREA_ITEMS_END_STR_ID	INTEGER	STR_ID_DOMAIN
BIO_END	INTEGER	
DIO_END	INTEGER	
PAGEFAULTS_END	INTEGER	
PAGEFAULT_IO_END	INTEGER	
CPU_END	INTEGER	
CURRENT_PRIO_END	SMALLINT	
VIRTUAL_SIZE_END	INTEGER	
WS_SIZE_END	INTEGER	
WS_PRIVATE_END	INTEGER	
WS_GLOBAL_END	INTEGER	
CROSS_FAC_2_END	INTEGER	
CROSS_FAC_3_END	INTEGER	
CROSS_FAC_7_END	INTEGER	
CROSS_FAC_14_END	INTEGER	

Table 6−7 shows the REQUEST_BLR table.

Table 6−7 Columns for Table EPC$1_221_REQUEST_BLR

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_POINT	DATE VMS	
CLIENT_PC	INTEGER	
STREAM_ID	INTEGER	
REQ_ID	INTEGER	
TRANS_ID	VARCHAR(16)	
TRANS_ID_STR_ID	INTEGER	STR_ID_DOMAIN
REQUEST_NAME	VARCHAR(31)	
REQUEST_NAME_STR_ID	INTEGER	STR_ID_DOMAIN
REQUEST_TYPE	INTEGER	
BLR	VARCHAR(127)	
BLR_STR_ID	INTEGER	STR_ID_DOMAIN

6.2 Address and Phone Number Correction for Documentation

In release 7.0 or earlier documentation, the address and fax phone number listed on the Send Us Your Comments page are incorrect. The correct information is:

```
FAX -- 603.897.3825
Oracle Corporation
One Oracle Drive
Nashua, NH 03062-2804
USA
```

6.3 Online Document Format and Ordering Information

You can view the documentation in Adobe Acrobat format using the Acrobat Reader, which allows anyone to view, navigate, and print documents in the Adobe Portable Document Format (PDF). See http://www.adobe.com for information about obtaining a free copy of Acrobat Reader and for information on supported platforms.

The Oracle Rdb documentation in Adobe Acrobat format is available on MetaLink:

```
Top Tech Docs\Oracle Rdb\Documentation\<bookname>
```

Customers should contact their Oracle representative to purchase printed documentation.

Chapter 7
Known Problems and Restrictions

This chapter describes problems and restrictions relating to Oracle Rdb and includes workarounds where appropriate.

7.1 Known Problems and Restrictions in All Interfaces

This section describes known problems and restrictions that affect all interfaces. This is not an exhaustive list. Check the Oracle Bug database to see a list of all open Rdb bugs and their current status.

7.1.1 Possible Incorrect Results When Using Partitioned Descending Indexes

Bug 6129797

In the current release of Oracle Rdb, 7.2.4, it is possible for some queries using partitioned indexes with segments of mixed ascending and descending order to return incorrect results either on Alpha or I64 systems.

The following examples show two problems when using partitioned index with segments of mixed ascending and descending order:

```
create database file foo
  create storage area fooa
  create storage area foob;

create table mesa (id integer, m4 char (1), m5 integer);
create table rasa (id integer, r4 char (1), r5 integer);

insert into mesa (id, m4, m5) values  (1, 'm', 1 );

insert into rasa (id, r4, r5) values  (1, 'm', 1 );
insert into rasa (id, r4, r5) values  (1, 'k', 1 );
insert into rasa (id, r4, r5) values  (1, 'e', 1 );

create index x4 on mesa (id asc , m4 asc) ;

! The following index contains ascending segments followed by descending
! segments and thus causes the query to return the wrong result.
!
! Note that the query works if both segments are either ascending or descending.
!
create index y4 on rasa (id asc , r4 desc)
       store using (id, r4)
       in fooa with limit of (1, 'g' )
       otherwise in foob ;
commit;

! Problem #1:
!
! the following query returns correctly 3 rows on Alpha but 1 row on IA64:

SQL> select m.id, m.m4, r.r4 from
   mesa m inner join rasa r on (m.id = r.id);
           1   m      m
           1   m      k
           1   m      e
3 rows selected

SQL> select m.id, m.m4, r.r4 from mesa m inner join rasa r on (m.id = r.id);
```

```
           1   m      e
1 row selected

! Problem #2:
!
! The following query using reverse scan returns 2 rows incorrectly on Alpha
! but 3 rows correctly on IA64:
!

SQL> select id, r4 from rasa where id = 1 and r4 <= 'm' order by id, r4;
Tables:
  0 = RASA
Index only retrieval of relation 0:RASA
  Index name  Y4 [2:1]   Reverse Scan
    Keys: (0.ID = 1) AND (0.R4 <= 'm')
       ID   R4
        1   k
        1   m
2 rows selected

SQL> select id, r4 from rasa where id = 1 and r4 <= 'm' order by id, r4;
Tables:
  0 = RASA
Index only retrieval of relation 0:RASA
  Index name  Y4 [2:1]   Reverse Scan
    Keys: (0.ID = 1) AND (0.R4 <= 'm')
       ID   R4
        1   e
        1   k
        1   m
3 rows selected
```

This problem is related to the construction and comparison of the descending key values during the index partitions.

The problem will be corrected in a future version of Oracle Rdb.

7.1.2 Remote Attach Stalls Before Detecting a Node is Unreachable

Bug 7681548

A remote attach can stall for a noticeable period, typically 75 seconds, before detecting a node is unreachable.

The following example shows the expected error message when attempting to access a database on a node that is not reachable. The problem is that when the value of the parameter SQL_NETWORK_TRANSPORT_TYPE in the file RDB$CLIENT_DEFAULTS.DAT is not specifically set to DECNET (in UPPER CASE), a stall of typically 75 seconds will happen before you get the expected error message.

```
SQL> attach 'file 1::disk1:[dbdir]db';
%SQL−F−ERRATTDEC, Error attaching to database 1::disk1:[dbdir]db
−RDB−F−IO_ERROR, input or output error
−SYSTEM−F−UNREACHABLE, remote node is not currently reachable
```

There are two possible ways to avoid the stall and get the error message after a user configurable period of time or instantly: decrease the value of the TCPIP parameter TCP_KEEPINIT, or explicitly specify SQL_NETWORK_TRANSPORT_TYPE as DECNET (in UPPER CASE).

- The default behavior when attempting to connect to an unreachable node via TCPIP is to stall 75 seconds before returning an error. The stall time is configurable in TCPIP via the parameter TCP_KEEPINIT which is expressed in units of 500 ms. The default value of TCP_KEEPINIT is 150 which corresponds to a 75 second stall.
- When connecting via DECnet, the error message is typically returned instantly so a significant stall will not be seen in this case. However, the value of the parameter SQL_NETWORK_TRANSPORT_TYPE is case sensitive so for DECnet to be selected as the transport, "DECNET" must be specified in UPPER CASE. Failing to do so will result in connecting via the DEFAULT method which is to first try connecting via DECnet and if that fails attempt to connect via TCPIP and hence a 75 second stall will take place unless TPC_KEEPINIT is set to a value lower than 150.

7.1.3 Case Sensitive Values in RDB$CLIENT_DEFAULTS.DAT

Bug 7681548

Various characteristics for network access to remote databases can be specified by entering parameters and values in a file named RDB$CLIENT_DEFAULTS.DAT. The following keywords that have character strings as their values only take effect if the values are specified in UPPER CASE: SQL_NETWORK_TRANSPORT_TYPE, SQL_MESSAGE_VECTOR_RETURN_TYPE, and SQL_DEFAULTS_RESTRICTION. The result of including one or more lower case characters in the value of one of these parameters is the same as if the parameter was not specified at all (for example, the default behavior would be applied and no error message would be issued).

In the following example, DECnet is specified with the last three characters in lower case. The result will be that the value of the parameter SQL_NETWORK_TRANSPORT_TYPE will be DEFAULT and not the intended value DECNET.

```
SQL_NETWORK_TRANSPORT_TYPE DECnet
```

This problem can be avoided by specifying the values for SQL_NETWORK_TRANSPORT_TYPE, SQL_MESSAGE_VECTOR_RETURN_TYPE, and SQL_DEFAULTS_RESTRICTION in RDB$CLIENT_DEFAULTS.DAT using UPPER CASE.

In the next major release of Oracle Rdb, the values in RDB$CLIENT_DEFAULTS.DAT will be case insensitive.

7.1.4 Standalone WITH Clause in Compound Statements Now Deprecated

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 recent syntax added to the ANSI and ISO SQL Database Language standard. Support for this new syntax (known as subquery factoring) is used by Oracle tools accessing Oracle Rdb through OCI Services for Rdb. Therefore, to accommodate this change Oracle will remove 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.

Any applications or SQL command files must be modified prior to the next major release of Oracle Rdb. At that time, the old syntax will be removed and the new WITH clause (aka subquery factoring) will be introduced.

The following example shows the old syntax, which now produces a deprecated message.

```
SQL> begin
cont> with hold preserve none
%SQL−I−DEPR_FEATURE, Deprecated Feature: WITH HOLD no longer supported in
this context − use PRAGMA (WITH HOLD) instead
cont> trace 'a';
cont> end;
```

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;
```

7.1.5 Calling DECC$CRTL_INIT

In cases where user−supplied code is being called by Oracle Rdb (such as an external function, a module called implementing the Oracle Backup API, or a user−supplied output routine for the Oracle Rdb LogMiner), if calls are made to certain DECC$SHR RTL routines, it may be required to first call DECC$CRTL_INIT.

DECC$CRTL_INIT is a C run time library routine that allows developers to call the HP C RTL from other languages or to use the HP C RTL when the main function is not in C. It initializes the run−time environment and establishes both an exit and condition handler. The Oracle Rdb main images are not written in C and should not be expected to have called DECC$CRTL_INIT prior to the user's code being invoked. The requirement for DECC$CRTL_INIT in certain cases exists in all versions of Oracle Rdb.

A shareable image need only call this function if it contains an HP C function call for signal handling, environment variables, I/O, exit handling, a default file protection mask, or if it is a child process that should inherit context. Although many of the initialization activities are performed only once, DECC$CRTL_INIT can safely be called multiple times.

See the HP C Run−Time Library Reference Manual for OpenVMS Systems manual for additional information.

7.1.6 Application and Oracle Rdb Both Using SYS$HIBER

In application processes that use Oracle Rdb and the SYS$HIBER system service (possibly via RTL routines such as LIB$WAIT), it is very important that the application ensures that the event being waited for has actually occurred. Oracle Rdb utilizes $HIBER/$WAKE sequences for interprocess communication and synchronization.

Because there is just a single process−wide "hibernate" state along with a single process−wide "wake pending" flag, Oracle Rdb must assume that it "shares" use of the hibernate/wake state with the user's application code. To this end, Oracle Rdb generally will re−wake the process via a pending wake request after using a hibernate sequence.

Oracle Rdb's use of the $WAKE system service will interfere with other users of $HIBER (such as the routine LIB$WAIT) that do not check for event completion, possibly causing a $HIBER to be unexpectedly resumed without waiting at all.

To avoid these situations, applications that use HIBER/WAKE facilities must use a code sequence that avoids continuing without a check for the operation (such as a delay or a timer firing) being complete.

The following pseudo−code shows one example of how a flag can be used to indicate that a timed−wait has completed correctly. The wait does not complete until the timer has actually fired and set TIMER_FLAG to TRUE. This code relies on ASTs being enabled.

```
ROUTINE TIMER_WAIT:
    BEGIN
    ! Clear the timer flag
    TIMER_FLAG = FALSE

    ! Schedule an AST for sometime in the future
    STAT = SYS$SETIMR (TIMADR = DELTATIME, ASTRTN = TIMER_AST)
    IF STAT <> SS$_NORMAL THEN LIB$SIGNAL (STAT)

    ! Hibernate.  When the $HIBER completes, check to make
    ! sure that TIMER_FLAG is set indicating that the wait
    ! has finished.
    WHILE TIMER_FLAG = FALSE
    DO  SYS$HIBER()
    END

ROUTINE TIMER_AST:
    BEGIN
    ! Set the flag indicating that the timer has expired
    TIMER_FLAG = TRUE

    ! Wake the main−line code
    STAT = SYS$WAKE ()
    IF STAT <> SS$_NORMAL THEN LIB$SIGNAL (STAT)
    END
```

Starting with OpenVMS V7.1, the LIB$WAIT routine includes a FLAGS argument (with the LIB$K_NOWAKE flag set) to allow an alternate wait scheme (using the $SYNCH system service) that can avoid potential problems with multiple code sequences using the $HIBER system service. See the OpenVMS RTL Library (LIB$) Manual for more information about the LIB$WAIT routine.

In order to prevent application hangs, inner−mode users of SYS$HIBER must take explicit steps to ensure that a pending wake is not errantly " consumed ". The general way of accomplishing this is to issue a SYS$WAKE to the process after the event is complete if a call to SYS$HIBER was done. Rdb takes this step and therefore application programs must be prepared for cases where a wakeup might appear unexpectedly.

7.1.7 Unexpected RCS Termination

It has been observed in internal testing of Rdb Release 7.2.2 that if the Record Cache Server (the RCS) terminates in an uncontrolled fashion this may, under some conditions, cause corruption of the database and/or the After Image Journal file.

When the RCS terminates, the database is shut down and a message like the following is written to the monitor log:

```
6−DEC−2007 15:04:17.02 − Received Record Cache Server image termination from
22ED5144:1
  − database name "device:[directory]database.RDB;1" [device] (1200,487,0)
  − abnormal Record Cache Server termination detected
  − starting delete−process shutdown of database:
    − %RDMS−F−RCSABORTED, record cache server process terminated abnormally
  − sending process deletion to process 22ED10F9
  − sending process deletion to process 22ECED59
  − sending process deletion to process 22EC0158
  − sending process deletion to process 22EB9543 (AIJ Log server)
  − database shutdown waiting for active users to terminate
```

A future attempt to roll forward the AIJ following a restore of a database backup might fail with a bugcheck dump if this problem has happened.

The only currently known situation where this problem has been observed is if the logical name RDM$BIND_RCS_VALIDATE_SECS is defined to some value and the logical name RDM$BIND_RCS_LOG_FILE at the same time is undefined or defined incorrectly.

To prevent this problem, Oracle recommends any customer using the Row Cache feature to either avoid defining the logical name RDM$BIND_RCS_VALIDATE_SECS or if this logical name for any reason needs to be defined, to make sure RDM$BIND_RCS_LOG_FILE is correctly defined (i.e. defined with the /SYSTEM and /EXECUTIVE qualifiers and is pointing to a valid file name in an existing directory on a cluster accessible device with sufficient free space).

This recommendation applies to all versions of Oracle Rdb.

7.1.8 Possible Incorrect Results When Using Partitioned Descending Indexes on I64

When running on I64 systems using Rdb Release 7.2, it is possible when using partitioned descending indexes for some queries to return incorrect results. Alpha systems are not effected by this problem.

The following example shows this difference in behavior between Alpha and I64 when using partitioned descending indexes:

```
SQL> CREATE DATABASE FILE FOO
cont>        CREATE STORAGE AREA FOOA
cont>        CREATE STORAGE AREA FOOB;
SQL>
SQL> CREATE TABLE MESA (ID INTEGER, M4 CHAR (1), M5 INTEGER);
SQL> CREATE TABLE RASA (ID INTEGER, R4 CHAR (1), R5 INTEGER);
SQL>
SQL> INSERT INTO MESA (ID, M4, M5) VALUES  (1, 'M', 1 );
```

```
1 row inserted
SQL> INSERT INTO RASA (ID, R4, R5) VALUES  (1, 'M', 1 );
1 row inserted
SQL>
SQL> CREATE INDEX X4 ON MESA (ID ASC , M4 DESC)
cont>        STORE USING (ID, M4)
cont>        IN FOOA WITH LIMIT OF (1, 'G')
cont>        OTHERWISE IN FOOB ;
SQL>
SQL> CREATE INDEX Y4 ON RASA (ID ASC , R4 DESC)
cont>        STORE USING (ID, R4)
cont>        IN FOOA WITH LIMIT OF (1, 'G' )
cont>        OTHERWISE IN FOOB ;
SQL>
SQL> COMMIT;

    ! This query correctly returns 1 row
    ! on Alpha but returns 0 rows on I64:

SQL> SELECT M.ID, M.M4, R.R4 FROM
cont> MESA M INNER JOIN RASA R ON (M.ID = R.ID);
0 rows selected
SQL>
```

This problem is related to the construction and comparison of the descending key values with Oracle Rdb running on I64. This problem will be corrected in a future Rdb 72 release.

7.1.9 Changes for Processing Existence Logical Names

This release of Oracle Rdb will change the handling of so called "existence" logical names used to tune the Rdb environment. These existence logical names could in past versions be defined to any value to enable their effect. The Rdb documentation in most cases described using the value 1 or YES as that value and this change is upward compatible with the documentation.

Rdb now treats these logical names (see the list below) as Boolean logicals and accepts a string starting with "Y", "y", "T", "t" or "1" to mean TRUE. All other values will be considered to be FALSE. This change allows process level definitions to override definitions in higher logical name tables which was not possible previously.

Oracle recommends that customers examine all procedures that define the following logical names to ensure that their values conform to these rules prior to upgrading to Oracle Rdb V7.2.1.1 or later to avoid unexpected changes in behavior.

- RDMS$AUTO_READY
- RDMS$DISABLE_HIDDEN_KEY
- RDMS$DISABLE_MAX_SOLUTION
- RDMS$DISABLE_REVERSE_SCAN
- RDMS$DISABLE_TRANSITIVITY
- RDMS$DISABLE_ZIGZAG_BOOLEAN
- RDMS$ENABLE_BITMAPPED_SCAN
- RDMS$ENABLE_INDEX_COLUMN_GROUP
- RDMS$MAX_STABILITY
- RDMS$USE_OLD_COST_MODEL
- RDMS$USE_OLD_COUNT_RELATION

- RDMS$USE_OLD_SEGMENTED_STRING
- RDMS$USE_OLD_UPDATE_RULES

7.1.10 Patch Required When Using VMS V8.3 and Dedicated CPU Lock Manager

During qualification testing of Oracle Rdb Release 7.2.1 on OpenVMS V8.3 systems, a problem with the use of Extended Lock Value Blocks and the OpenVMS Dedicated CPU Lock Manager feature was discovered.

To avoid this problem, Oracle strongly recommends that customers wishing to use Oracle Rdb and the OpenVMS Dedicated CPU Lock Manager feature with OpenVMS V8.3 install one of the following architecture−specific patch kit (or subsequent replacement if superseded) prior to using Oracle Rdb Release 7.2.1 on OpenVMS V8.3 systems:

- VMS83I_SYS−V0200 (I64)
- VMS83A_SYS−V0100 (Alpha)

7.1.11 SQL Module or Program Fails with %SQL−F−IGNCASE_BAD

Bug 2351258

A SQL Module or Pre−compiled SQL program built with Rdb 6.1 or earlier may fail when running under Rdb 7.2 if the program submits queries that involve certain kinds of character operations on parameters in the queries. For example, a LIKE operator in the WHERE clause of a SQL statement requires SQL to look for character− or string−matching wildcard characters. Another example is the use of IGNORE CASE which causes SQL to equivalence upper and lower case characters for the character set in use.

The following example shows a portion of a SQL module language program that queries a PERSONNEL database.

```
DECLARE MANL_NAME_LIST CURSOR FOR
 SELECT DISTINCT E.LAST_NAME,E.FIRST_NAME,J.JOB_CODE,J.DEPARTMENT_CODE,E.CITY
FROM   DB1_HANDLE.EMPLOYEES E,DB1_HANDLE.JOB_HISTORY J
 WHERE J.EMPLOYEE_ID = E.EMPLOYEE_ID
   AND E.STATUS_CODE = STATUS_CODE
   AND E.CITY LIKE CITYKEY IGNORE CASE
   ORDER BY E.EMPLOYEE_ID DESC, E.LAST_NAME DESC

PROCEDURE SQL_OPN_NAME_LIST
SQLCODE
CITYKEY       CHAR(20)
STATUS_CODE   CHAR(1);
OPEN MANL_NAME_LIST;
```

If the SQL Module containing the code above is compiled and linked into an executable using a pre−7.0 version of Rdb, it will run properly against that version. However if the same program is run in an Rdb 7.2 environment, a call to the SQL_OPN_NAME_LIST procedure will return a SQLCODE of −1. The RDB$MESSAGE_VECTOR will contain a code associated with the following message:

```
%SQL−F−IGNCASE_BAD, IGNORE CASE not supported for character set
```

To workaround this problem, re−link the program using a 7.2 version of SQL$INT.EXE and/or SQL$USER.OLB.

7.1.12 External Routine Images Linked with PTHREAD$RTL

The OpenVMS Guide to the POSIX Threads Library describes that it is not supported to dynamically activate the core run−time library shareable image PTHREAD$RTL. Oracle has found in testing that a shareable image supplied for use as an External Routine that is linked with PTHREAD$RTL can be expected to cause a hang during dynamic image activation on OpenVMS I64 systems. This problem has not been observed on OpenVMS Alpha systems.

To avoid this problem in any case where the shareable image used for an Rdb External Routine is linked with PTHREAD$RTL, the main program image must likewise be linked with PTHREAD$RTL. This requirement applies to customer built application main programs as well as the main interactive SQL image.

The shareable image RDB$NATCONN_FUNC72.EXE supplied with OCI Services for Oracle Rdb (part of SQL/Services) is one such shareable image that is linked with PTHREAD$RTL. Customer built applications that utilize External Routines from the RDB$NATCONN_FUNC72.EXE image must ensure that the main image is linked with PTHREAD$RTL. The external routines that a user may call that use functions from RDB$NATCONN_FUNC72.EXE include:

- TO_CHAR
- TO_NUMBER
- TO_DATE

You can use the OpenVMS command ANALYZE/IMAGE to determine whether an image depends upon PTHREAD$RTL. For more information, see the OpenVMS documentation.

7.1.13 Using Databases from Releases Earlier than V7.0

You cannot convert or restore databases earlier than the Oracle Rdb V7.0 format directly to Oracle Rdb V7.2 format. The RMU Convert command for Oracle Rdb V7.2 supports conversions from Oracle Rdb V7.0 and V7.1 format databases only. If you have an Oracle Rdb V3.0 through V6.1 format database, you must convert it to at least Oracle Rdb V7.0 format and then convert it to Oracle Rdb V7.2 format. For example, if you have a V4.2 format database, you must convert it first to at least Oracle Rdb V7.0 format, then convert it to Oracle Rdb V7.2 format.

If you attempt to convert or restore a database that is prior to Oracle Rdb V7.0 format directly to Oracle Rdb V7.2 format, Oracle RMU generates an error.

7.1.14 Partitioned Index with Descending Column and Collating Sequence

Bug 2797443

A known problem exists in which a query can return wrong results (number of rows returned is incorrect). This can happen on a table that has a multi−column, partitioned index in which one of the columns is sorted in descending order and the column has an associated collating sequence.

The following example can be used to demonstrate the problem.

```
$ sql$
create database file mf_collating.rdb alloc 10
  collating sequence french french
  create storage area area1 alloc 10
  create storage area area2 alloc 10
  create storage area area3 alloc 10;
create table tab1 (id tinyint, r3 char (3));
insert into tab1 (id, r3) values (1, 'a');
insert into tab1 (id, r3) values (1, 'b');
insert into tab1 (id, r3) values (1, 'f');
create index y3 on tab1 (id asc, r3 desc)
       store using (id, r3)
       in area1 with limit of (1, 'k')
       in area2 with limit of (1, 'e')
       otherwise in area3 ;
commit;

set flags 'strategy';

! Here is a query that returns the correct rows using sequential rather
! than indexed access.

select id, r3 from tab1 where id = 1 and r3 <= 'e'
  optimize for sequential access;
Conjunct        Get     Retrieval sequentially of relation TAB1
   ID   R3
    1   a
    1   b
2 rows selected

! Here is the same query without the sequential access restriction.
! Note in the query strategy that index Y3 is used for data retrieval.
! This query ought to (but does not) return the same set of rows as
! for the sequential access query.

select id, r3 from tab1 where id = 1 and r3 <= 'e';
Leaf#01 FFirst TAB1 Card=3
  BgrNdx1 Y3 [2:1] Fan=16
0 rows selected
```

7.1.15 Domain−Qualified TCP/IP Node Names in Distributed Transactions

Bug 3735144

When using TCP/IP for Oracle Rdb remote connections, distributed transactions involving databases on nodes which are not on the same subnet may not work.

Remote Rdb has the capability to make remote connections via TCP/IP in lieu of DECnet. (See the Oracle Rdb OpenVMS Installation and Configuration Guide for how to set this up.) However, distributed transactions involving remote databases connected to via TCP/IP have been difficult. This is because Rdb relies on OpenVMS DECdtm for distributed transaction support and DECdtm requires DECnet for off−node communication. (This is an OpenVMS and not an Rdb restriction. Contact Hewlett−Packard OpenVMS Support for more details.)

OpenVMS provides a capability to run DECnet over TCP/IP so that OpenVMS services which require DECnet (like DECdtm) can operate in an environment where a TCP/IP network is used as the communications backbone. This capability allows DECdtm (and hence Rdb) to manage distributed transactions via TCP/IP. (See HP's OpenVMS DECnet−Plus documentation set for how to configure and use this capability.)

However, for a transaction involving a remote database, Rdb only provides the SCSNODE name of the remote node to DECdtm. For example, consider the following SQL attaches to two remote databases using TCP/IP:

```
SQL> attach 'alias db1 filename node1.a.b.c::db_root:db1 user ''me'' using
''pw''';
SQL> attach 'alias db2 filename node1.a.b.c::db_root:db2 user ''me'' using
''pw''';
```

In the above example, Rdb can successfully connect to both remote databases using the TCP/IP address "node1.a.b.c." but when multiple databases are attached, Rdb implicitly uses distributed transactions via DECdtm. Since Rdb only passes DECdtm the SCSNODE name retrieved from the RDBSERVERnn at the other end of the connection, DECdtm does not, in general, have the information it needs to resolve the remote reference. It will only be able to do so if the SCSNODE name and the TCP/IP node name are the same and the local node is on the same subnet (i.e. ".a.b.c" in the example). Otherwise, after the second attach is made, the following error message will be received as soon as a transaction is started:

```
SQL> set trans read write;
%RDB−F−SYS_REQUEST_CAL, error from system services request − called from 100001
−RDB−E−DECDTMERR, DECdtm system service call error
−IPC−E−BCKTRNSFAIL, failure on the back translate address request
```

There are three potential workarounds:

- If distributed transactions are unimportant to the application, they can be disabled by defining the logical name SQL$DISABLE_CONTEXT to TRUE. Rdb will then not call DECdtm and the node name resolution problem will not be seen. However, it will be the problem of the application to maintain database integrity in the event that a commit succeeds on one database and not on another. See the Rdb Guide to Distributed Transactions for more information.
- If all the nodes involved in the distributed transaction are in the same domain, then TCP/IP can resolve the node with only the first part of the node provided that the SCSNODE name is identical to it. In the example above, this would mean that the remote node had an SCSNODE name of "NODE1" and that the local node was on TCP/IP subnet ".a.b.c".
- It may also be possible to define a DNS/BIND alias name for the remote node's SCSNODE name to the local node's TCP/IP database. This should allow the SCSNODE name passed by Rdb Dispatch to be translated successfully. For example, assuming HP TCP/IP Services for OpenVMS is the TCP/IP protocol stack then a command like the following could be used on the local node:

  ```
  $ TCP SET HOST NODE1.A.B.C/address=nnn.nnn.nnn.nnn/alias=NODE1_SCS
  ```

  Where "nnn.nnn.nnn.nnn" is the IP address and "NODE1_SC" the OpenVMS SCSNODE name of the remote node. See the HP DECnet−Plus documentation set for more information on how to maintain

TCP/IP domain databases.

7.1.16 ILINK−E−INVOVRINI Error on I64

When linking an application with multiple modules, the following error message may be returned:

```
%ILINK−E−INVOVRINI, incompatible multiple initializations for overlaid section
        section: VMSRDB
        module: M1
        file: DKA0:[BLD]M1.OBJ;1
        module: M2
        file: DKA0:[BLD]SYS.OLB;1
```

On I64 systems, it is not allowed to have a program section that attempts to be initialized a subsequent time where the non−zero portions of the initializations do not match. This is a difference from OpenVMS Alpha and VAX systems where the linker permitted such initializations.

If the modules specified are SQL module language or precompiler produced, the application build procedures usually need to be modified. Typically, the solution is to initialize the database handles in only one of the modules. The SQLMOD command line qualifiers /NOINITIALIZE_HANDLES and /INITIALIZE_HANDLES are used to specify whether or not alias definitions are coerced into alias references.

7.1.17 New Attributes Saved by RMU/LOAD Incompatible With Prior Versions

Bug 2676851

To improve the behavior of unloading views, Oracle Rdb Release 7.1.2 changed the way view columns were unloaded so that attributes for view computed columns, COMPUTED BY and AUTOMATIC columns were saved. These new attributes are not accepted by prior releases of Oracle Rdb.

The following example shows the reported error trying to load a file from V7.1.2 under V7.1.0.4.

```
%RMU−F−NOTUNLFIL, Input file was not created by RMU UNLOAD
%RMU−I−DATRECSTO,   0 data records stored.
%RMU−F−FTL_LOAD, Fatal error for LOAD operation at 21−OCT−2003 16:34:54.20
```

You can workaround this problem by using the /RECORD_DEFINITION qualifier and specifying the FORMAT=DELIMITED option. However, this technique does not support LIST OF BYTE VARYING column unloading.

7.1.18 SYSTEM−F−INSFMEM Fatal Error With SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED in Galaxy Environment

When using the GALAXY SUPPORT IS ENABLED feature in an OpenVMS Galaxy environment, a *%SYSTEM−F−INSFMEM, insufficient dynamic memory error* may be returned when mapping record caches or opening the database. One source of this problem specific to a Galaxy configuration is running out of Galaxy Shared Memory regions. For Galaxy systems, GLX_SHM_REG is the number of shared memory region structures configured into the Galaxy Management Database (GMDB).

While the default value (for OpenVMS versions through at least V7.3−1) of 64 regions might be adequate for some installations, sites using a larger number of databases or row caches when the SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED features are enabled may find the default insufficient.

If a *%SYSTEM−F−INSFMEM, insufficient dynamic memory* error is returned when mapping record caches or opening databases, Oracle Corporation recommends that you increase the GLX_SHM_REG parameter by 2 times the sum of the number of row caches and number of databases that might be accessed in the Galaxy at one time. As the Galaxy shared memory region structures are not very large, setting this parameter to a higher than required value does not consume a significant amount of physical memory. It also may avoid a later reboot of the Galaxy environment. This parameter must be set on all nodes in the Galaxy.

Galaxy Reboot Required

Changing the GLX_SHM_REG system parameter requires that the OpenVMS Galaxy environment be booted from scratch. That is, all nodes in the Galaxy must be shut down and then the Galaxy reformed by starting each instance.

7.1.19 Oracle Rdb and OpenVMS ODS−5 Volumes

OpenVMS Version 7.2 introduced an Extended File Specifications feature, which consists of two major components:

- A new, optional, volume structure, ODS−5, which provides support for file names that are longer and have a greater range of legal characters than in previous versions of OpenVMS.
- Support for "deep" directory trees.

ODS−5 was introduced primarily to provide enhanced file sharing capabilities for users of Advanced Server for OpenVMS 7.2 (formerly known as PATHWORKS for OpenVMS), as well as DCOM and JAVA applications.

In some cases, Oracle Rdb performs its own file and directory name parsing and explicitly requires ODS−2 (the traditional OpenVMS volume structure) file and directory name conventions to be followed. Because of this knowledge, Oracle does not support any Oracle Rdb database file components (including root files, storage area files, after image journal files, record cache backing store files, database backup files, after image journal backup files, etc.) that utilize any non−ODS−2 file naming features. For this reason, Oracle recommends that Oracle Rdb database components not be located on ODS−5 volumes.

Oracle does support Oracle Rdb database file components on ODS−5 volumes provided that all of these files and directories used by Oracle Rdb strictly follow the ODS−2 file and directory name conventions. In particular, all file names must be specified entirely in uppercase and "special" characters in file or directory names are forbidden.

7.1.20 Optimization of Check Constraints

Bug 1448422

When phrasing constraints using the "CHECK" syntax, a poorer strategy can be chosen by the optimizer than when the same or similar constraint is phrased using referential integrity (PRIMARY and FOREIGN KEY) constraints.

For example, I have two tables T1 and T2, both with one column, and I wish to ensure that all values in table T1 exist in T2. Both tables have an index on the referenced field. I could use a PRIMARY KEY constraint on T2 and a FOREIGN KEY constraint on T1.

```
SQL> alter table t2 alter column f2 primary key not deferrable;
SQL> alter table t1 alter column f1 references t2 not deferrable;
```

When deleting from the PRIMARY KEY table, Rdb will only check for rows in the FOREIGN KEY table where the FOREIGN KEY has the deleted value. This can be seen as an index lookup on T1 in the retrieval strategy.

```
SQL> delete from t2 where f2=1;
Get     Temporary relation      Retrieval by index of relation T2
  Index name  I2 [1:1]
Index only retrieval of relation T1
  Index name  I1 [1:1]
%RDB−E−INTEG_FAIL, violation of constraint T1_FOREIGN1 caused operation to fail
```

The failure of the constraint is not important. What is important is that Rdb efficiently detects that only those rows in T1 with the same values as the deleted row in T2 can be affected.

It is necessary sometimes to define this type of relationship using CHECK constraints. This could be necessary because the presence of NULL values in the table T2 precludes the definition of a primary key on that table. This could be done with a CHECK constraint of the form:

```
SQL> alter table t1 alter column f1
cont>   check (f1 in (select * from t2)) not deferrable;
SQL> delete from t2 where f2=1;
Get     Temporary relation      Retrieval by index of relation T2
  Index name  I2 [1:1]
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation T1
      Index name  I1 [0:0]
  Cross block entry 2
    Conjunct        Aggregate−F1    Conjunct
    Index only retrieval of relation T2
      Index name  I2 [0:0]
%RDB−E−INTEG_FAIL, violation of constraint T1_CHECK1 caused operation to fail
```

The cross block is for the constraint evaluation. This retrieval strategy indicates that to evaluate the constraint, the entire index on table T1 is being scanned and for each key, the entire index in table T2 is being scanned. The behavior can be improved somewhat by using an equality join condition in the select clause of the constraint:

```
SQL> alter table t1 alter column f1
cont>   check (f1 in (select * from t2 where f2=f1)) not deferrable;
```

```
or:

SQL> alter table t1 alter column f1
cont>   check (f1=(select * from t2 where f2=f1)) not deferrable;
```

In both cases the retrieval strategy will look like this:

```
SQL> delete from t2 where f2=1;
Get     Temporary relation      Retrieval by index of relation T2
  Index name  I2 [1:1]
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation T1
      Index name  I1 [0:0]
  Cross block entry 2
    Conjunct        Aggregate−F1    Conjunct
    Index only retrieval of relation T2
      Index name  I2 [1:1]
%RDB−E−INTEG_FAIL, violation of constraint T1_CHECK1 caused operation to fail
```

While the entire T1 index is scanned, at least the value from T1 is used to perform an index lookup on T2.

These restrictions result from semantic differences in the behavior of the "IN" and "EXISTS" operators with respect to null handling, and the complexity of dealing with non−equality join conditions.

To improve the performance of this type of integrity check on larger tables, it is possible to use a series of triggers to perform the constraint check. The following triggers perform a similar check to the constraints above.

```
SQL> create trigger t1_insert after insert on t1
cont>  when (not exists (select * from t2 where f2=f1))
cont>    (error) for each row;
SQL> create trigger t1_update after update on t1
cont>  when (not exists (select * from t2 where f2=f1))
cont>    (error) for each row;
SQL> ! A delete trigger is not needed on T1.
SQL> create trigger t2_delete before delete on t2
cont>  when (exists (select * from t1 where f1=f2))
cont>    (error) for each row;
SQL> create trigger t2_modify after update on t2
cont>  referencing old as t2o new as t2n
cont>  when (exists (select * from t1 where f1=t2o.f2))
cont>    (error) for each row;
SQL> ! An insert trigger is not needed on T2.
```

The strategy for a delete on T2 is now:

```
SQL> delete from t2 where f2=1;
Aggregate−F1    Index only retrieval of relation T1
  Index name  I1 [1:1]
Temporary relation      Get     Retrieval by index of relation T2
  Index name  I2 [1:1]
%RDB−E−TRIG_INV_UPD, invalid update; encountered error condition defined for
trigger
−RDMS−E−TRIG_ERROR, trigger T2_DELETE forced an error
```

The trigger strategy is the index only retrieval displayed first. You will note that the index on T1 is used to examine only those rows that may be affected by the delete.

Care must be taken when using this workaround as there are semantic differences in the operation of the triggers, the use of "IN" and "EXISTS", and the use of referential integrity constraints.

This workaround is useful where the form of the constraint is more complex, and cannot be phrased using referential integrity constraints. For example, if the application is such that the value in table T1 may be spaces or NULL to indicate the absence of a value, the above triggers could easily be modified to allow for these semantics.

7.1.21 Carryover Locks and NOWAIT Transaction Clarification

In NOWAIT transactions, the BLAST (Blocking AST) mechanism cannot be used. For the blocking user to receive the BLAST signal, the requesting user must request the locked resource with WAIT (which a NOWAIT transaction does not do). Oracle Rdb defines a resource called NOWAIT, which is used to indicate that a NOWAIT transaction has been started. When a NOWAIT transaction starts, the user requests the NOWAIT resource. All other database users hold a lock on the NOWAIT resource so that when the NOWAIT transaction starts, all other users are notified with a NOWAIT BLAST. The BLAST causes blocking users to release any carryover locks. There can be a delay before the transactions with carryover locks detect the presence of the NOWAIT transaction and release their carryover locks. You can detect this condition by examining the stall messages. If the "Waiting for NOWAIT signal (CW)" stall message appears frequently, the application is probably experiencing a decrease in performance, and you should consider disabling the carryover lock behavior.

7.1.22 Unexpected Results Occur During Read−Only Transactions on a Hot Standby Database

When using Hot Standby, it is typical to use the standby database for reporting, simple queries, and other read−only transactions. If you are performing these types of read−only transactions on a standby database, be sure you can tolerate a READ COMMIT level of isolation. This is because the Hot Standby database might be updated by another transaction before the read−only transaction finishes, and the data retrieved might not be what you expected.

Because Hot Standby does not write to the snapshot files, the isolation level achieved on the standby database for any read−only transaction is a READ COMMITED transaction. This means that nonrepeatable reads and phantom reads are allowed during the read−only transaction:

- Nonrepeatable read operations: Allows the return of different results within a single transaction when an SQL operation reads the same row in a table twice. Nonrepeatable reads can occur when another transaction modifies and commits a change to the row between transactions. Because the standby database will update the data when it confirms a transaction has been committed, it is very possible to see an SQL operation on a standby database return different results.
- Phantom read operations: Allows the return of different results within a single transaction when an SQL operation retrieves a range of data values (or similar data existence check) twice. Phantoms can occur if another transaction inserted a new record and committed the insertion between executions of the range retrieval. Again, because the standby database may do this, phantom reads are possible.

Thus, you cannot rely on any data read from the standby database to remain unchanged. Be sure your read−only transactions can tolerate a READ COMMIT level of isolation before you implement procedures that read and use data from a standby database.

7.1.23 Row Cache Not Allowed While Hot Standby Replication is Active

The row cache feature may not be enabled on a hot standby database while replication is active. The hot standby feature will not start if row cache is enabled.

This restriction exists because rows in the row cache are accessed via logical dbkeys. However, information transferred to the standby database via the after image journal facility only contains physical dbkeys. Because there is no way to maintain rows in the cache via the hot standby processing, the row cache must be disabled when the standby database is open and replication is active.

A new command qualifier, ROW_CACHE=DISABLED, has been added to the RMU Open command. To open the hot standby database prior to starting replication, use the ROW_CACHE=DISABLED qualifier on the RMU Open command.

7.1.24 Excessive Process Page Faults and Other Performance Considerations During Oracle Rdb Sorts

Excessive hard or soft page faulting can be a limiting factor of process performance. One factor contributing to Oracle Rdb process page faulting is sorting operations. Common causes of sorts include the SQL GROUP BY, ORDER BY, UNION, and DISTINCT clauses specified for a query, and index creation operations. Defining the logical name RDMS$DEBUG_FLAGS to "RS" can help determine when Oracle Rdb sort operations are occurring and to display the sort keys and statistics.

Oracle Rdb includes its own copy of the OpenVMS SORT32 code within the Oracle Rdb images and does not generally call the routines in the OpenVMS run−time library. A copy of the SORT32 code is used to provide stability between versions of Oracle Rdb and OpenVMS and because Oracle Rdb calls the sort routines from executive processor mode which is difficult to do using the SORT32 shareable image. SQL IMPORT and RMU Load operations do, however, call the OpenVMS SORT run−time library.

At the beginning of a sort operation, the SORT code allocates memory for working space. The SORT code uses this space for buffers, in−memory copies of the data, and sorting trees.

SORT does not directly consider the processes quotas or parameters when allocating memory. The effects of WSQUOTA and WSEXTENT are indirect. At the beginning of each sort operation, the SORT code attempts to adjust the process working set to the maximum possible size using the $ADJWSL system service specifying a requested working set limit of %X7FFFFFFF pages (the maximum possible). SORT then uses a value of 75% of the returned working set for virtual memory scratch space. The scratch space is then initialized and the sort begins.

The initialization of the scratch space generally causes page faults to access the pages newly added to the working set. Pages that were in the working set already may be faulted out as the new pages are faulted in. Once the sort operation completes and SORT returns back to Oracle Rdb, the pages that may have been faulted out of the working set are likely to be faulted back into the working set.

When a process working set is limited by the working set quota (WSQUOTA) parameter and the working set extent (WSEXTENT) parameter is a much larger value, the first call to the sort routines can cause many page faults as the working set grows. Using a value of WSEXTENT that is closer to WSQUOTA can help reduce the impact of this case.

With some OpenVMS versions, AUTOGEN sets the SYSGEN parameter PQL_MWSEXTENT equal to the WSMAX parameter. This means that all processes on the system end up with WSEXTENT the same as WSMAX. Since that might be quite high, sorting might result in excessive page faulting. You may want to explicitly set PQL_MWSEXTENT to a lower value if this is the case on your system.

Sort work files are another factor to consider when tuning for Oracle Rdb sort operations. When the operation can not be done in the available memory, SORT uses temporary disk files to hold the data as it is being sorted. The Oracle Rdb7 Guide to Database Performance and Tuning contains more detailed information about sort work files.

The logical name RDMS$BIND_SORT_WORKFILES specifies how many work files sort is to use if work files are required. The default is 2 and the maximum number is 36. The work files can be individually controlled by the SORTWORKn logical names (where n ranges from "0" through "Z"). You can increase the efficiency of sort operations by assigning the location of the temporary sort work files to different disks. These assignments are made by using up to 36 logical names, "SORTWORK0" through "SORTWORKZ".

Normally, SORT places work files in the your SYS$SCRATCH directory. By default, SYS$SCRATCH is the same device and directory as the SYS$LOGIN location. Spreading the I/O load over multiple disks and/or controllers improves efficiency as well as performance by taking advantage of more system resources and helps prevent disk I/O bottlenecks. Specifying that a your work files reside on separate disks permits overlap of the SORT read/write cycle. You may also encounter cases where insufficient space exists on the SYS$SCRATCH disk device (for example, while Oracle Rdb builds indexes for a very large table). Using the "SORTWORK0" through "SORTWORKZ" logical names can help you avoid this problem.

Note that SORT uses the work files for different sorted runs, and then merges the sorted runs into larger groups. If the source data is mostly sorted, then not every sort work file may need to be accessed. This is a possible source of confusion because even with 36 sort work files, it is possible to exceed the capacity of the first SORT file device and the sort operation fails never having accessed the remaining 35 sort work files.

At this time, more than 10 sort work files will only be used by the Oracle Rdb sort interface as used by the CREATE INDEX, ALTER INDEX and the clauses UNION DISTINCT, ORDER BY, GROUP BY and SELECT DISTINCT. The RMU and SQL IMPORT interfaces use the OpenVMS SORT interface which does not currently support more than 10 sort work files.

Note that the logical names RDMS$BIND_WORK_VM and RDMS$BIND_WORK_FILE do not affect or control the operation of sort. These logical names are used to control other temporary space allocation within Oracle Rdb.

7.1.25 Control of Sort Work Memory Allocation

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

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

The logical names are:

Table 7−1 Sort Memory Logicals

Logical	Definition
RDMS$BIND_SORT_MEMORY_WS_FACTOR	Specifies a percentage of the process's working set limit to be used when allocating sort memory for the built−in SORT32 package. If not defined, the default value is 75 (representing 75%), the maximum value is 75 (representing 75%), and the minimum value is 2 (representing 2%). Processes with vary large working set limits can sometimes experience significant page faulting and CPU consumption while initializing sort memory. This logical name can restrict the sort work memory to a percentage of the processes maximum working set.
RDMS$BIND_SORT_MEMORY_MAX_BYTES	Specifies an absolute limit to be used when allocating sort memory for the built−in SORT32 package. If not defined, the default value is unlimited (up to 1GB), the maximum value is 2147483647 and the minimum value is 32768.

7.1.26 The Halloween Problem

When a cursor is processing rows selected from a table, it is possible that another separate query can interfere with the retrieval of the cursor by modifying the index columns key values used by the cursor.

For instance, if a cursor selects all EMPLOYEES with LAST_NAME >= 'M', it is likely that the query will use the sorted index on LAST_NAME to retrieve the rows for the cursor. If an update occurs during the processing of the cursor which changes the LAST_NAME of an employee from "Mason" to "Rickard", then it is possible that that employee row will be processed twice. First when it is fetched with name "Mason", and then later when it is accessed by the new name "Rickard".

The Halloween problem is a well known problem in relational databases. Access strategies which optimize the I/O requirements, such as Index Retrieval, can be subject to this problem. Interference from queries by other sessions are avoided by locking and are controlled by the ISOLATION LEVEL options in SQL, or the CONCURRENCY/CONSISTENCY options in RDO/RDML.

Oracle Rdb avoids this problem if it knows that the cursors subject table will be updated. For example, if the SQL syntax UPDATE ... WHERE CURRENT OF is used to perform updates of target rows, or the RDO/RDML MODIFY statement uses the context variable for the stream. Then the optimizer will choose an alternate access strategy if an update can occur which may cause the Halloween problem. This can be seen in the access strategy in Example 2−2 as a "Temporary relation" being created to hold the result of the cursor query.

When you use interactive or dynamic SQL, the UPDATE ... WHERE CURRENT OF or DELETE ... WHERE CURRENT OF statements will not be seen until after the cursor is declared and opened. In these environments, you must use the FOR UPDATE clause to specify that columns selected by the cursor will be updated during cursor processing. This is an indication to the Rdb optimizer so that it protects against the Halloween problem in this case. This is shown in Example 2−1 and Example 2−2.

The following example shows that the EMP_LAST_NAME index is used for retrieval. Any update performed will possibly be subject to the Halloween problem.

```
SQL> set flags 'strategy';
SQL> declare emp  cursor for
cont> select * from employees where last_name >= 'M' order by last_name;
SQL> open emp;
Conjunct        Get     Retrieval by index of relation EMPLOYEES
  Index name  EMP_LAST_NAME [1:0]
SQL> close emp;
```

The following example shows that the query specifies that the column LAST_NAME will be updated by some later query. Now the optimizer protects the EMP_LAST_NAME index used for retrieval by using a "Temporary Relation" to hold the query result set. Any update performed on LAST_NAME will now avoid the Halloween problem.

```
SQL> set flags 'strategy';
SQL> declare emp2 cursor for
cont> select * from employees where last_name >= 'M'
cont> order by last_name for update of last_name;
SQL> open emp2;
Temporary relation      Conjunct        Get
Retrieval by index of relation EMPLOYEES
  Index name  EMP_LAST_NAME [1:0]
SQL> close emp2;
```

When you use the SQL precompiler, or the SQL module language compiler it can be determined from usage that the cursor context will possibly be updated during the processing of the cursor because all cursor related statements are present within the module. This is also true for the RDML/RDBPRE precompilers when you use the DECLARE_STREAM and START_STREAM statements and use the same stream context to perform all MODIFY and ERASE statements.

The point to note here is that the protection takes place during the open of the SQL cursor (or RDO stream), not during the subsequent UPDATE or DELETE.

If you execute a separate UPDATE query which modifies rows being fetched from the cursor then the actual rows fetched will depend upon the access strategy chosen by the Rdb optimizer. As the query is separate from the cursors query (i.e. doesn't reference the cursor context), then the optimizer does not know that the cursor selected rows are potentially updated and so cannot perform the normal protection against the Halloween problem.

7.2 SQL Known Problems and Restrictions

This section describes known problems and restrictions for the SQL interface.

7.2.1 SET FLAGS CRONO_FLAG Removed

The SET FLAGS statement and RDMS$SET_FLAGS logical name no longer accept the obsolete keyword CRONO_FLAG. This keyword has been removed. Please update all scripts and applications to use the keyword CHRONO_FLAG.

7.2.2 Interchange File (RBR) Created by Oracle Rdb Release 7.2 Not Compatible With Previous Releases

To support the large number of new database attributes and objects, the protocol used by SQL EXPORT and SQL IMPORT has been enhanced to support more protocol types. Therefore, this format of the Oracle Rdb release 7.2 interchange files can no longer be read by older versions of Oracle Rdb.

Oracle Rdb continues to provide upward compatibility for interchange files generated by older versions.

Oracle Rdb has never supported backward compatibility, however, it was sometimes possible to use an interchange file with an older version of IMPORT. However, this protocol change will no longer permit this usage.

7.2.3 Single Statement LOCK TABLE is Not Supported for SQL Module Language and SQL Precompiler

The new LOCK TABLE statement is not currently supported as a single statement within the module language or embedded SQL language compiler.

Instead you must enclose the statement in a compound statement. That is, use BEGIN... END around the statement as shown in the following example. This format provides all the syntax and flexibility of LOCK TABLE.

This restriction does not apply to interactive or dynamic SQL.

The following extract from the module language listing file shows the reported error if you use LOCK TABLE as a single statement procedure. The other procedure in the same module is acceptable because it uses a compound statement that contains the LOCK TABLE statement.

```
1 MODULE sample_test
2 LANGUAGE C
3 PARAMETER COLONS
4
5 DECLARE ALIAS FILENAME 'mf_personnel'
6
7 PROCEDURE a (SQLCODE);
8 LOCK TABLE employees FOR EXCLUSIVE WRITE MODE;
%SQL−F−WISH_LIST, (1) Feature not yet implemented − LOCK TABLE requires compound
statement
```

```

PROCEDURE b (SQLCODE);
BEGIN
LOCK TABLE employees FOR EXCLUSIVE WRITE MODE;
END;
```

To workaround this problem of using LOCK TABLE for SQL module language or embedded SQL application, use a compound statement in an EXEC SQL statement.

7.2.4 Multistatement or Stored Procedures May Cause Hangs

Long−running multistatement or stored procedures can cause other users in the database to hang if the procedures obtain resources needed by those other users. Some resources obtained by the execution of a multistatement or stored procedure are not released until the multistatement or stored procedure finishes. Thus, any−long running multistatement or stored procedure can cause other processes to hang. This problem can be encountered even if the statement contains SQL COMMIT or ROLLBACK statements.

The following example demonstrates the problem. The first session enters an endless loop; the second session attempts to backup the database but hangs forever.

```
Session 1:

SQL> attach 'filename MF_PERSONNEL';
SQL> create function LIB$WAIT (in real by reference)
cont> returns integer;
cont> external name LIB$WAIT location 'SYS$SHARE:LIBRTL.EXE'
cont> language general general parameter style variant;
SQL> commit;
        .
        .
        .
$ SQL
SQL> attach 'filename MF_PERSONNEL';
SQL> begin
cont> declare :LAST_NAME LAST_NAME_DOM;
cont> declare :WAIT_STATUS integer;
cont> loop
cont> select LAST_NAME into :LAST_NAME
cont> from EMPLOYEES where EMPLOYEE_ID = '00164';
cont> rollback;
cont> set :WAIT_STATUS = LIBWAIT (5.0);
cont> set transaction read only;
cont> end loop;
cont> end;

Session 2:

$ RMU/BACKUP/LOG/ONLINE MF_PERSONNEL MF_PERSONNEL

From a third session, you can see that the backup process is waiting for a lock
held in the first session:

$ RMU/SHOW LOCKS /MODE=BLOCKING MF_PERSONNEL
    .
    .
    .
```

```
Resource: nowait signal

ProcessID Process Name        Lock ID  System ID Requested Granted
--------- ---------------     --------- -------- --------- -------
20204383  RMU BACKUP.....     5600A476  00010001 CW        NL
2020437B  SQL............     3B00A35C  00010001 PR        PR
```

There is no workaround for this restriction. When the multistatement or stored procedure finishes execution, the resources needed by other processes are released.

7.2.5 Use of Oracle Rdb from Shareable Images

If code in the image initialization routine of a shareable image makes any calls into Oracle Rdb, through SQL or any other means, access violations or other unexpected behavior may occur if Oracle Rdb images have not had a chance to do their own initialization.

To avoid this problem, applications must take one of the following steps:

- Do not make Oracle Rdb calls from the initialization routines of shareable images.
- Link in such a way that the RDBSHR.EXE image initializes first. You can do this by placing the reference to RDBSHR.EXE and any other Oracle Rdb shareable images last in the linker options file.

This is not a bug; it is a restriction resulting from the way OpenVMS image activation works.

7.3 Oracle RMU Known Problems and Restrictions

This section describes known problems and restrictions for the RMU interface.

7.3.1 RMU Convert Fails When Maximum Relation ID is Exceeded

If, when relation IDs are assigned to new system tables during an RMU Convert to a V7.2 database, the maximum relation ID of 8192 allowed by Oracle Rdb is exceeded, the fatal error %RMU−F−RELMAXIDBAD is displayed and the database is rolled back to the prior database version. Contact your Oracle support representative if you get this error. Note that when the database is rolled back, the fatal error %RMU−F−CVTROLSUC is displayed to indicate that the rollback was successful but caused by the detection of a fatal error and not requested by the user.

This condition only occurs if there are an extremely large number of tables defined in the database or if a large number of tables were defined but have subsequently been deleted.

The following example shows both the %RMU−F−RELMAXIDBAD error message if the allowed database relation ID maximum of 8192 is exceeded and the %RMU−F−CVTROLSUC error message when the database has been rolled back to V7.0 since it cannot be converted to V7.2:

```
$rmu/convert mf_personnel
%RMU−I−RMUTXT_000, Executing RMU for Oracle Rdb V7.2
Are you satisfied with your backup of
 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 and your backup of
 any associated .aij files [N]? Y
 %RMU−I−LOGCONVRT, database root converted to current structure level
 %RMU−F−RELMAXIDBAD, ROLLING BACK CONVERSION − Relation ID exceeds maximum
  8192 for system table RDB$RELATIONS
 %RMU−F−CVTROLSUC, CONVERT rolled−back for
  DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 to version V7.0
```

The following example shows the normal case when the maximum allowed relation ID is not exceeded:

```
$rmu/convert mf_personnel
%RMU−I−RMUTXT_000, Executing RMU for Oracle Rdb V7.2
Are you satisfied with your backup of
 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 and your backup of
 any associated .aij files [N]? Y
%RMU−I−LOGCONVRT, database root converted to current structure level
%RMU−S−CVTDBSUC, database DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
 successfully converted from version V7.0 to V7.2
%RMU−I−CVTCOMSUC, CONVERT committed for
 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 to version V7.2
```

7.3.2 RMU Unload /After_Journal Requires Accurate AIP Logical Area Information

The RMU Unload /After_Journal command uses the on−disk area inventory pages (AIPs) to determine the appropriate type of each logical area when reconstructing logical dbkeys for records stored in mixed−format storage areas. However, the logical area type information in the AIP is generally unknown for logical areas created prior to Oracle Rdb release 7.0.1. If the RMU Unload /After_Journal command cannot determine the logical area type for one or more AIP entries, a warning message is displayed for each such area and may ultimately return logical dbkeys with a 0 (zero) area number for records stored in mixed−format storage areas.

In order to update the on−disk logical area type in the AIP, the RMU Repair utility must be used. The INITIALIZE=LAREA_PARAMETERS=optionfile qualifier option file can be used with the TYPE qualifier. For example, to repair the EMPLOYEES table of the MF_PERSONNEL database, you would create an options file that contains the following line:

```
EMPLOYEES /TYPE=TABLE
```

For partitioned logical areas, the AREA=name qualifier can be used to identify the specific storage areas that are to be updated. For example, to repair the EMPLOYEES table of the MF_PERSONNEL database for the EMPID_OVER storage area only, you would create an options file that contains the following line:

```
EMPLOYEES /AREA=EMPID_OVER /TYPE=TABLE
```

The TYPE qualifier specifies the type of a logical area. The following keywords are allowed:

- TABLE
  Specifies that the logical area is a data table. This would be a table created using the SQL CREATE TABLE syntax.
- B−TREE
  Specifies that the logical area is a B−tree index. This would be an index created using the SQL CREATE INDEX TYPE IS SORTED syntax.
- HASH
  Specifies that the logical area is a hash index. This would be an index created using the SQL CREATE INDEX TYPE IS HASHED syntax.
- SYSTEM
  Specifies that the logical area is a system record that is used to identify hash buckets. Users cannot explicitly create these types of logical areas.

  Note

  This type should NOT be used for the RDB$SYSTEM logical areas. This type does NOT identify system relations.
- BLOB
  Specifies that the logical area is a BLOB repository.

There is no explicit error checking of the type specified for a logical area. However, an incorrect type may cause the RMU Unload /After_Journal command to be unable to correctly return valid, logical dbkeys.

7.3.3 Do Not Use HYPERSORT with RMU Optimize After_Journal Command

The OpenVMS Alpha V7.1 operating system introduced the high−performance Sort/Merge utility (also known as HYPERSORT). This utility takes advantage of the OpenVMS Alpha architecture to provide better performance for most sort and merge operations.

The high−performance Sort/Merge utility supports a subset of the SOR routines. Unfortunately, the high−performance Sort/Merge utility does not support several of the interfaces used by the RMU Optimize After_Journal command. In addition, the high−performance Sort/Merge utility reports no error or warning when being called with the unsupported options used by the RMU Optimize After_Journal command.

Because of this, the use of the high−performance Sort/Merge utility is not supported for the RMU Optimize After_Journal command. Do not define the logical name SORTSHR to reference HYPERSORT.EXE.

7.3.4 Changes in EXCLUDE and INCLUDE Qualifiers for RMU Backup

The RMU Backup command no longer accepts both the Include and Exclude qualifiers in the same command. This change removes the confusion over exactly what gets backed up when Include and Exclude are specified on the same line, but does not diminish the capabilities of the RMU Backup command.

To explicitly exclude some storage areas from a backup, use the Exclude qualifier to name the storage areas to be excluded. This causes all storage areas to be backed up except for those named by the Exclude qualifier.

Similarly, the Include qualifier causes only those storage areas named by the qualifier to be backed up. Any storage area not named by the Include qualifier is not backed up. The Noread_only and Noworm qualifiers continue to cause read−only storage areas and WORM storage areas to be omitted from the backup even if these areas are explicitly listed by the Include qualifier.

Another related change is in the behavior of EXCLUDE=*. In previous versions, EXCLUDE=* caused all storage areas to be backed up. Beginning with V7.1, EXCLUDE=* causes only a root backup to be done. A backup created by using EXCLUDE=* can be used only by the RMU Restore Only_Root command.

7.3.5 RMU Backup Operations Should Use Only One Type of Tape Drive

When using more than one tape drive for an RMU Backup command, all of the tape drives must be of the same type (for example, all the tape drives must be TA90s or TZ87s or TK50s). Using different tape drive types (for example, one TK50 and one TA90) for a single database backup operation may make database restoration difficult or impossible.

Oracle RMU attempts to prevent using different tape drive densities during a backup operation, but is not able to detect all invalid cases and expects that all tape drives for a backup are of the same type.

As long as all of the tapes used during a backup operation can be read by the same type of tape drive during a restore operation, the backup is likely valid. This may be the case, for example, when using a TA90 and a TA90E.

Oracle Corporation recommends that, on a regular basis, you test your backup and recovery procedures and environment using a test system. You should restore the database and then recover using AIJs to simulate failure recovery of the production system.

Consult the Oracle Rdb7 Guide to Database Maintenance, the Oracle Rdb7 Guide to Database Design and Definition, and the Oracle RMU Reference Manual for additional information about Oracle Rdb backup and restore operations.

7.3.6 RMU/VERIFY Reports PGSPAMENT or PGSPMCLST Errors

RMU/VERIFY may sometimes report PGSPAMENT or PGSPMCLST errors when verifying storage areas. These errors indicate that the Space Area Management (SPAM) page fullness threshold for a particular data page does not match the actual space usage on the data page. For a further discussion of SPAM pages, consult the Oracle Rdb7 Guide to Database Maintenance.

In general, these errors will not cause any adverse affect on the operation of the database. There is potential for space on the data page to not be totally utilized, or for a small amount of extra I/O to be expended when searching for space in which to store new rows. But unless there are many of these errors then the impact should be negligible.

It is possible for these inconsistencies to be introduced by errors in Oracle Rdb. When those cases are discovered, Oracle Rdb is corrected to prevent the introduction of the inconsistencies. It is also possible for these errors to be introduced during the normal operation of Oracle Rdb. The following scenario can leave the SPAM pages inconsistent:

1. A process inserts a row on a page, and updates the threshold entry on the corresponding SPAM page to reflect the new space utilization of the data page. The data page and SPAM pages are not flushed to disk.
2. Another process notifies the first process that it would like to access the SPAM page being held by the process. The first process flushes the SPAM page changes to disk and releases the page. Note that it has not flushed the data page.
3. The first process then terminates abnormally (for example, from the DCL STOP/IDENTIFICATION command). Since that process never flushed the data page to disk, it never wrote the changes to the Recovery Unit Journal (RUJ) file. Since there were no changes in the RUJ file for that data page then the Database Recovery (DBR) process did not need to roll back any changes to the page. The SPAM page retains the threshold update change made above even though the data page was never flushed to disk.

While it would be possible to create mechanisms to ensure that SPAM pages do not become out of synch with their corresponding data pages, the performance impact would not be trivial. Since these errors are relatively rare and the impact is not significant, then the introduction of these errors is considered to be part of the normal operation of Oracle Rdb. If it can be proven that the errors are not due to the scenario above, then Oracle Product Support should be contacted.

PGSPAMENT and PGSPMCLST errors may be corrected by doing any one of the following operations:

- Recreate the database by performing:
    1. SQL EXPORT
    2. SQL DROP DATABASE
    3. SQL IMPORT
- Recreate the database by performing:
    1. RMU/BACKUP
    2. SQL DROP DATABASE

3. RMU/RESTORE

* Repair the SPAM pages by using the RMU/REPAIR command. Note that the RMU/REPAIR command does not write its changes to an after−image journal (AIJ) file. Therefore, Oracle recommends that a full database backup be performed immediately after using the RMU/REPAIR command.

7.4 Known Problems and Restrictions in All Interfaces for Release 7.0 and Earlier

The following problems and restrictions from release 7.0 and earlier still exist.

7.4.1 Converting Single−File Databases

Because of a substantial increase in the database root file information for V7.0, you should ensure that you have adequate disk space before you use the RMU Convert command with single−file databases and V7.0 or higher.

The size of the database root file of any given database increases a maximum of about 600 disk blocks. The actual increase depends mostly on the maximum number of users specified for the database.

7.4.2 Row Caches and Exclusive Access

If a table has a row−level cache defined for it, the Row Cache Server (RCS) may acquire a shared lock on the table and prevent any other user from acquiring a Protective or Exclusive lock on that table.

7.4.3 Exclusive Access Transactions May Deadlock with RCS Process

If a table is frequently accessed by long running transactions that request READ/WRITE access reserving the table for EXCLUSIVE WRITE and if the table has one or more indexes, you may experience deadlocks between the user process and the Row Cache Server (RCS) process.

There are at least three suggested workarounds to this problem:

- Reserve the table for SHARED WRITE
- Close the database and disable row cache for the duration of the exclusive transaction
- Change the checkpoint interval for the RCS process to a time longer than the time required to complete the batch job and then trigger a checkpoint just before the batch job starts. Set the interval back to a smaller interval after the checkpoint completes.

7.4.4 Strict Partitioning May Scan Extra Partitions

When you use a WHERE clause with the less than (<) or greater than (>) operator and a value that is the same as the boundary value of a storage map, Oracle Rdb scans extra partitions. A boundary value is a value specified in the WITH LIMIT OF clause. The following example, executed while the logical name RDMS$DEBUG_FLAGS is defined as "S", illustrates the behavior:

```
ATTACH 'FILENAME MF_PERSONNEL';
CREATE TABLE T1 (ID INTEGER, LAST_NAME CHAR(12), FIRST_NAME CHAR(12));
CREATE STORAGE MAP M FOR T1 PARTITIONING NOT UPDATABLE
 STORE USING (ID)
 IN EMPIDS_LOW WITH LIMIT OF (200)
 IN EMPIDS_MID WITH LIMIT OF (400)
 OTHERWISE IN EMPIDS_OVER;
INSERT INTO T1 VALUES (150,'Boney','MaryJean');
INSERT INTO T1 VALUES (350,'Morley','Steven');
INSERT INTO T1 VALUES (300,'Martinez','Nancy');
INSERT INTO T1 VALUES (450,'Gentile','Russ');
SELECT * FROM T1 WHERE ID > 400;
Conjunct Get Retrieval sequentially of relation T1
Strict Partitioning: part 2 3
ID LAST_NAME FIRST_NAME
450 Gentile Russ
1 row selected
```

In the previous example, partition 2 does not need to be scanned. This does not affect the correctness of the result. Users can avoid the extra scan by using values other than the boundary values.

7.4.5 Restriction When Adding Storage Areas with Users Attached to Database

If you try to interactively add a new storage area where the page size is less than the smallest existing page size and the database has been manually opened or users are active, the add operation fails with the following errors:

```
%RDMS−F−NOEUACCESS, unable to acquire exclusive access to database
```

or

```
%RDB−F−SYS_REQUEST, error from system services request
−RDMS−F−FILACCERR, error opening database root DKA0:[RDB]TEST.RDB;1
−SYSTEM−W−ACCONFLICT, file access conflict
```

You can make this change only when no users are attached to the database and, if the database is set to OPEN IS MANUAL, the database is closed. Several internal Oracle Rdb data structures are based on the minimum page size and these structures cannot be resized if users are attached to the database.

Furthermore, because this particular change is not recorded in the AIJ, any recovery scenario fails. Note also that if you use .aij files, you must backup the database and restart after−image journaling because this change invalidates the current AIJ recovery.

7.4.6 Multiblock Page Writes May Require Restore Operation

If a node fails while a multiblock page is being written to disk, the page in the disk becomes inconsistent, and is detected immediately during failover. (Failover is the recovery of an application by restarting it on another computer.) The problem is rare, and occurs because only single−block I/O operations are guaranteed by OpenVMS to be written atomically. This problem has never been reported by any customer and was detected only during stress tests in our labs.

Correct the page by an area−level restore operation. Database integrity is not compromised, but the affected area is not available until the restore operation completes.

A future release of Oracle Rdb will provide a solution that guarantees multiblock atomic write operations. Cluster failovers will automatically cause the recovery of multiblock pages, and no manual intervention will be required.

7.4.7 Replication Option Copy Processes Do Not Process Database Pages Ahead of an Application

When a group of copy processes initiated by the Replication Option (formerly Data Distributor) begins running after an application has begun modifying the database, the copy processes catch up to the application and are not able to process database pages that are logically ahead of the application in the RDB$CHANGES system relation. The copy processes all align waiting for the same database page and do not move on until the application has released it. The performance of each copy process degrades because it is being paced by the application.

When a copy process completes updates to its respective remote database, it updates the RDB$TRANSFERS system relation and then tries to delete any RDB$CHANGES rows not needed by any transfers. During this process, the RDB$CHANGES table cannot be updated by any application process, holding up any database updates until the deletion process is complete. The application stalls while waiting for the RDB$CHANGES table. The resulting contention for RDB$CHANGES SPAM pages and data pages severely impacts performance throughput, requiring user intervention with normal processing.

This is a known restriction in V4.0 and higher. Oracle Rdb uses page locks as latches. These latches are held only for the duration of an action on the page and not to the end of transaction. The page locks also have blocking asynchronous system traps (ASTs) associated with them. Therefore, whenever a process requests a page lock, the process holding that page lock is sent a blocking AST (BLAST) by OpenVMS. The process that receives such a blocking AST queues the fact that the page lock should be released as soon as possible. However, the page lock cannot be released immediately.

Such work requests to release page locks are handled at verb commit time. An Oracle Rdb verb is an Oracle Rdb query that executes atomically, within a transaction. Therefore, verbs that require the scan of a large table, for example, can be quite long. An updating application does not release page locks until its verb has completed.

The reasons for holding on to the page locks until the end of the verb are fundamental to the database management system.

7.5 SQL Known Problems and Restrictions for Oracle Rdb Release 7.0 and Earlier

The following problems and restrictions from Oracle Rdb Release 7.0 and earlier still exist.

7.5.1 ARITH_EXCEPT or Incorrect Results Using LIKE IGNORE CASE

When you use LIKE...IGNORE CASE, programs linked under Oracle Rdb V4.2 and V5.1, but run under higher versions of Oracle Rdb, may result in incorrect results or %RDB−E−ARITH_EXCEPT exceptions.

To work around the problem, avoid using IGNORE CASE with LIKE or recompile and relink under a higher version (V6.0 or higher.)

7.5.2 Different Methods of Limiting Returned Rows from Queries

You can establish the query governor for rows returned from a query by using either the SQL SET QUERY LIMIT statement or a logical name. This note describes the differences between the two mechanisms.

If you define the RDMS$BIND_QG_REC_LIMIT logical name to a small value, the query often fails with no rows returned regardless of the value assigned to the logical. The following example demonstrates setting the limit to 10 rows and the resulting failure:

```
$ DEFINE RDMS$BIND_QG_REC_LIMIT 10
$ SQL$
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES;
%RDB−F−EXQUOTA, Oracle Rdb runtime quota exceeded
−RDMS−E−MAXRECLIM, query governor maximum limit of rows has been reached
```

Interactive SQL must load its metadata cache for the table before it can process the SELECT statement. In this example, interactive SQL loads its metadata cache to allow it to check that the column EMPLOYEE_ID really exists for the table. The queries on the Oracle Rdb system relations RDB$RELATIONS and RDB$RELATION_FIELDS exceed the limit of rows.

Oracle Rdb does not prepare the SELECT statement, let alone execute it. Raising the limit to a number less than 100 (the cardinality of EMPLOYEES) but more than the number of columns in EMPLOYEES (that is, the number of rows to read from the RDB$RELATION_FIELDS system relation) is sufficient to read each column definition.

To see an indication of the queries executed against the system relations, define the RDMS$DEBUG_FLAGS logical name as "S" or "B".

If you set the row limit using the SQL SET QUERY statement and run the same query, it returns the number of rows specified by the SQL SET QUERY statement before failing:

```
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SET QUERY LIMIT ROWS 10;
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES;
EMPLOYEE_ID
00164
00165
   .
   .
   .
00173
%RDB−E−EXQUOTA, Oracle Rdb runtime quota exceeded
−RDMS−E−MAXRECLIM, query governor maximum limit of rows has been reached
```

The SET QUERY LIMIT specifies that only user queries be limited to 10 rows. Therefore, the queries used to load the metadata cache are not restricted in any way.

Like the SET QUERY LIMIT statement, the SQL precompiler and module processor command line qualifiers (QUERY_MAX_ROWS and SQLOPTIONS=QUERY_MAX_ROWS) only limit user queries.

Keep the differences in mind when limiting returned rows using the logical name RDMS$BIND_QG_REC_LIMIT. They may limit more queries than are obvious. This is important when using 4GL tools, the SQL precompiler, the SQL module processor, and other interfaces that read the Oracle Rdb system relations as part of query processing.

7.5.3 Suggestions for Optimal Use of SHARED DATA DEFINITION Clause for Parallel Index Creation

The CREATE INDEX process involves the following steps:

1. Process the metadata.
2. Lock the index name.
   Because new metadata (which includes the index name) is not written to disk until the end of the index process, Oracle Rdb must ensure index name uniqueness across the database during this time by taking a special lock on the provided index name.
3. Read the table for sorting by selected index columns and ordering.
4. Sort the key data.
5. Build the index (includes partitioning across storage areas).
6. Write new metadata to disk.

Step 6 is the point of conflict with other index definers because the system relation and indexes are locked like any other updated table.

Multiple users can create indexes on the same table by using the RESERVING table_name FOR SHARED DATA DEFINITION clause of the SET TRANSACTION statement. For optimal usage of this capability, Oracle Rdb suggests the following guidelines:

- You should commit the transaction immediately after the CREATE INDEX statement so that locks on the table are released. This avoids lock conflicts with other index definers and improves overall concurrency.
- By assigning the location of the temporary sort work files SORTWORK0, SORTWORK1, ... , SORTWORK9 to different disks for each parallel process that issues the SHARED DATA DEFINITION statement, you can increase the efficiency of sort operations. This minimizes

any possible disk I/O bottlenecks and allows overlap of the SORT read/write cycle.

- ♦ If possible, enable global buffers and specify a buffer number large enough to hold a sufficient amount of table data. However, do not define global buffers larger than the available system physical memory. Global buffers allow sharing of database pages and thus result in disk I/O savings. That is, pages are read from disk by one of the processes and then shared by the other index definers for the same table, reducing the I/O load on the table.
- ♦ If global buffers are not used, ensure that enough local buffers exist to keep much of the index cached (use the RDM$BIND_BUFFERS logical name or the NUMBER OF BUFFERS IS clause in SQL to change the number of buffers).
- ♦ To distribute the disk I/O load, store the storage areas for the indexes on separate disk drives. Note that using the same storage area for multiple indexes results in contention during the index creation (Step 5) for SPAM pages.
- ♦ Consider placing the .ruj file for each parallel definer on its own disk or an infrequently used disk.
- ♦ Even though snapshot I/O should be minimal, consider disabling snapshots during parallel index creation.
- ♦ Refer to the Oracle Rdb7 Guide to Database Performance and Tuning to determine the appropriate working set values for each process to minimize excessive paging activity. In particular, avoid using working set parameters where the difference between WSQUOTA and WSEXTENT is large. The SORT utility uses the difference between these two values to allocate scratch virtual memory. A large difference (that is, the requested virtual memory grossly exceeds the available physical memory) may lead to excessive page faulting.
- ♦ The performance benefits of using SHARED DATA DEFINITION can best be observed when creating many indexes in parallel. The benefit is in the average elapsed time, not in CPU or I/O usage. For example, when two indexes are created in parallel using the SHARED DATA DEFINITION clause, the database must be attached twice, and the two attaches each use separate system resources.
- ♦ Using the SHARED DATA DEFINITION clause on a single−file database or for indexes defined in the RDB$SYSTEM storage area is not recommended.

The following table displays the elapsed time benefit when creating multiple indexes in parallel with the SHARED DATA DEFINITION clause. The table shows the elapsed time for ten parallel process index creations (Index1, Index2, ... Index10) and one process with ten sequential index creations (All10). In this example, global buffers are enabled and the number of buffers is 500. The longest time for a parallel index creation is Index7 with an elapsed time of 00:02:34.64, compared to creating ten indexes sequentially with an elapsed time of 00:03:26.66. The longest single parallel create index elapsed time is shorter than the elapsed time of creating all ten of the indexes serially.

Table 7−2 Elapsed Time for Index Creations

Index Create Job	Elapsed Time
Index1	00:02:22.50
Index2	00:01:57.94
Index3	00:02:06.27
Index4	00:01:34.53
Index5	00:01:51.96
Index6	00:01:27.57
Index7	00:02:34.64
Index8	00:01:40.56

Index9	00:01:34.43
Index10	00:01:47.44
All10	00:03:26.66

7.5.4 Side Effect When Calling Stored Routines

When calling a stored routine, you must not use the same routine to calculate argument values by a stored function. For example, if the routine being called is also called by a stored function during the calculation of an argument value, passed arguments to the routine may be incorrect.

The following example shows a stored procedure P being called during the calculation of the arguments for another invocation of the stored procedure P:

```
SQL> create module M
cont>     language SQL
cont>
cont>     procedure P (in :a integer, in :b integer, out :c integer);
cont>     begin
cont>     set :c = :a + :b;
cont>     end;
cont>
cont>     function F () returns integer
cont>     comment is 'expect F to always return 2';
cont>     begin
cont>     declare :b integer;
cont>     call P (1, 1, :b);
cont>     trace 'returning ', :b;
cont>     return :b;
cont>     end;
cont> end module;
SQL>
SQL> set flags 'TRACE';
SQL> begin
cont> declare :cc integer;
cont> call P (2, F(), :cc);
cont> trace 'Expected 4, got ', :cc;
cont> end;
~Xt: returning 2
~Xt: Expected 4, got 3
```

The result as shown above is incorrect. The routine argument values are written to the called routine's parameter area before complex expression values are calculated. These calculations may (as in the example) overwrite previously copied data.

The workaround is to assign the argument expression (in this example calling the stored function F) to a temporary variable and pass this variable as the input for the routine. The following example shows the workaround:

```
SQL> begin
cont> declare :bb, :cc integer;
cont> set :bb = F();
cont> call P (2, :bb, :cc);
cont> trace 'Expected 4, got ', :cc;
cont> end;
~Xt: returning 2
```

```
~Xt: Expected 4, got 4
```

This problem will be corrected in a future version of Oracle Rdb.

7.5.5 Considerations When Using Holdable Cursors

If your applications use holdable cursors, be aware that after a COMMIT or ROLLBACK statement is executed, the result set selected by the cursor may not remain stable. That is, rows may be inserted, updated, and deleted by other users because no locks are held on the rows selected by the holdable cursor after a commit or rollback occurs. Moreover, depending on the access strategy, rows not yet fetched may change before Oracle Rdb actually fetches them.

As a result, you may see the following anomalies when using holdable cursors in a concurrent user environment:

- ♦ If the access strategy forces Oracle Rdb to take a data snapshot, the data read and cached may be stale by the time the cursor fetches the data.
  For example, user 1 opens a cursor and commits the transaction. User 2 deletes rows read by user 1 (this is possible because the read locks are released). It is possible for user 1 to report data now deleted and committed.
- ♦ If the access strategy uses indexes that allow duplicates, updates to the duplicates chain may cause rows to be skipped, or even revisited.
  Oracle Rdb keeps track of the dbkey in the duplicate chain pointing to the data that was fetched. However, the duplicates chain could be revised by the time Oracle Rdb returns to using it.

Holdable cursors are a very powerful feature for read−only or predominantly read−only environments. However, in concurrent update environments, the instability of the cursor may not be acceptable. The stability of holdable cursors for update environments will be addressed in future versions of Oracle Rdb.

You can define the logical name RDMS$BIND_HOLD_CURSOR_SNAP to the value 1 to force all hold cursors to fetch the result set into a cached data area. (The cached data area appears as a "Temporary Relation" in the optimizer strategy displayed by the SET FLAGS 'STRATEGY' statement or the RDMS$DEBUG_FLAGS "S" flag.) This logical name helps to stabilize the cursor to some degree.

7.5.6 AIJSERVER Privileges

For security reasons, the AIJSERVER account ("RDMAIJSERVER") is created with only NETMBX and TMPMBX privileges. These privileges are sufficient to start Hot Standby, in most cases.

However, for production Hot Standby systems, these privileges are not adequate to ensure continued replication in all environments and workload situations. Therefore, Oracle recommends that the DBA provide the following additional privileges for the AIJSERVER account:

- ♦ ALTPRI − This privilege allows the AIJSERVER to adjust its own priority to ensure adequate quorum (CPU utilization) to prompt message processing.
- ♦ PSWAPM − This privilege allows the AIJSERVER to enable and disable process swapping, also necessary to ensure prompt message processing.

- ♦ SETPRV − This privilege allows the AIJSERVER to temporarily set any additional privileges it may need to access the standby database or its server processes.
- ♦ SYSPRV − This privilege allows the AIJSERVER to access the standby database rootfile, if necessary.
- ♦ WORLD − This privilege allows the AIJSERVER to more accurately detect standby database server process failure and handle network failure more reliably.

| Contents