Oracle® Rdb for OpenVMS

Release Notes

Release 7.4.1.0

July 2020

ORACLE®

Oracle Rdb Release Notes, Release 7.4.1.0 for OpenVMS

Primary Author: Rdb Engineering and Documentation group

Contents

Preface vii

1 Installing Oracle Rdb Release 7.4.1.0

1.1 Oracle Rdb on OpenVMS Industry Standard 64 1–1
1.2 Requirements 1–1
1.2.1 Ensure No Processes Have RDMSHRP Image Activated 1–2
1.3 Deprecated Features 1–2
1.3.1 Deprecated Support for the PL/I Compiler 1–2
1.4 Intel Itanium Processor 9700 “Kittson” Certified 1–3
1.5 Hardware Emulation Support 1–3
1.6 Maximum OpenVMS Version Check 1–3
1.7 Database Format Changed 1–3
1.8 Using Databases from Releases Earlier than V7.0 1–3
1.9 Changes to Installation Procedure 1–4
1.10 Invoking the VMSINSTAL Procedure 1–5
1.11 Stopping the Installation 1–5
1.12 After Installing Oracle Rdb 1–6
1.13 VMS$MEM_RESIDENT_USER Rights Identifier Required 1–6

2 Software Errors Fixed in Oracle Rdb Release 7.4.1.0

2.1 Software Errors Fixed That Apply to All Interfaces 2–1
2.1.1 RDMS-F-BADPAGNUM Exception After Snapshot File Truncated . . . 2–1
2.1.2 SQL/Services V7.4 IVP Fails With RMU-E-NOSQSCLIENT 2–1
2.1.3 Long Running Delete Does Not Advance After-Image Journal Checkpoint 2–2
2.1.4 Long Running Query Blocks Other Queries 2–2
2.2 SQL Errors Fixed 2–3
2.2.1 Unexpected Bugcheck When Using DROP STORAGE AREA ... CASCADE 2–3
2.2.2 Unexpected Error %SQL-F-FLDNOTCRS For Declaring a LIST Cursor 2–3
2.3 RMU Errors Fixed 2–3
2.3.1 Clarification of RMU Unload After_Image Header 2–3
2.3.2 Unexpected RMU/UNLOAD/AFTER Bugcheck at AIJEXT$QSORT_RELEASE_REC 2–4

3 Enhancements And Changes Provided in Oracle Rdb Release 7.4.1.0

3.1 Enhancements And Changes Provided in Oracle Rdb Release 7.4.1.0 3–1
3.1.1 New PCSI Support for Rdb Kit Installation and Deinstallation 3–1
3.1.2 Some Aggregate Functions Now Inherit Source Column EDIT STRING 3–2
3.1.3 Enhanced LIKE Table Support in CREATE TABLE Statement 3–3
3.1.4 Comma Statement Separator in Trigger Body No Longer Supported 3–4
3.1.5 New RMU RECLAIM /FREE_PAGES Qualifier Frees Unused Data Page Clumps 3–4
3.1.6 CREATE DEFAULT AUDIT Now Supports CREATE OR REPLACE Syntax and Semantics 3–9
3.1.7 New System Privileges Feature 3–10
3.1.8 New Database Vault Feature 3–20
3.1.9 Support for VSI TCP/IP for OpenVMS Version 10.6 or Later 3–21
3.1.10 New SET FLAGS Keyword for Hash Join Feature - HASHING 3–22
3.1.11 New JOIN BY HASH Clause in CREATE OUTLINE Statement 3–23
3.1.12 New Hash Join Feature 3–24
3.1.13 New ALTER DATABASE ... LOAD ACL IDENTIFIERS Clause 3–26
3.1.14 New ALTER TABLE Actions for READ ONLY Table 3–28
3.1.15 New NULLS FIRST and NULLS LAST Options for ORDER BY Clause 3–29
3.1.16 Enhancements to RMU Unload After_Image (LogMiner) Interface . . . 3–30
3.1.16.1 New XML Option to FORMAT Qualifier 3–30
3.1.16.2 TRIM Option 3–31
3.1.16.3 SYMBOLS Qualifier 3–31
3.1.17 New Named Partition Support for RESERVING Clause 3–32
3.1.18 RMU Backup No Longer Supports HUFFMAN or LZSS Compression, Use ZLIB Instead 3–33

4 Documentation Corrections, Additions and Changes

4.1 Documentation Corrections 4–1
4.1.1 Oracle Rdb Release 7.4.x.x New Features Document Added 4–1
4.1.2 New Optional Builtin Function RDB$$IS_ROW_FRAGMENTED 4–1
4.1.3 Undocumented /TRANSACTION=EXCLUSIVE Option for RMU Populate_Cache 4–2
4.1.4 Action of DECLARE TRANSACTION When SQL$DATABASE is Defined 4–3
4.1.5 RDB$USAGE Field Values 4–3
4.1.6 Updated Documentation for RDMS$BIND_CODE_OPTIMIZATION Logical 4–4
4.1.7 New RDM$BIND_AIJ_BUFFER_POOL_COUNT Logical 4–5
4.1.8 RDMSTT Image Optionally Installed 4–7
4.1.9 Some Optional System Tables Can Be Relocated Using a User Defined Storage Map 4–7
4.1.10 Recovering an Oracle Rdb Database After RMU/RESTORE/ONLY_ROOT 4–9
4.1.11 Oracle Rdb Position on NFS Devices 4–10
4.1.12 RDM$BIND_STAREA_EMERGENCY_DIR Logical Name 4–11
4.1.13 RDMS-F-FULLAIJBKUP, Partially-Journaled Changes Made 4–12
4.1.14 Undocumented Hot Standby Logical Names 4–14
4.1.15 Missing Documentation for the TRANSACTION_TYPE Keyword for GET DIAGNOSTICS 4–16

4.1.16 Corrections to the EDIT STRING Documentation 4–17
4.1.17 Changes and Improvements to the Rdb Optimizer and Query Compiler 4–18
4.1.18 Sorting Capabilities in Oracle Rdb 4–21
4.1.19 RDM$BIND_MAX_DBR_COUNT Documentation Clarification 4–22
4.1.20 Missing Tables Descriptions for the RDBEXPERT Collection Class . . . 4–23
4.1.21 Missing Columns Descriptions for Tables in the Formatted Database 4–24
4.2 RDO, RDBPRE and RDB$INTERPRET Features 4–32
4.2.1 New Request Options for RDO, RDBPRE and RDB$INTERPRET 4–32
4.2.2 New Language Features for RDO and Rdb Precompiler 4–34
4.2.3 RDO Interface Now Supports Synonym References 4–36
4.3 Address and Phone Number Correction for Documentation 4–37
4.4 Online Document Format and Ordering Information 4–37

5 Known Problems and Restrictions

5.1 Known Problems and Restrictions in All Interfaces 5–1
5.1.1 The Format of the RMU/ANALYZE/BINARY_OUTPUT Files Can Change 5–1
5.1.2 RMU /VERIFY /KEY_VALUES May Fail on Some Indices 5–2
5.1.3 REPLACE Statement Fails With Primary Key Constraint Failure When Used on a View 5–3
5.1.4 Possible Incorrect Results When Using Partitioned Descending Indexes 5–3
5.1.5 Application and Oracle Rdb Both Using SYS$HIBER 5–5
5.1.6 Unexpected RCS Termination 5–6
5.1.7 External Routine Images Linked with PTHREAD$RTL 5–6
5.1.8 ILINK-E-INVOVRINI Error on I64 5–7
5.1.9 SYSTEM-F-INSFMEM Fatal Error With SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED in Galaxy Environment 5–7
5.1.10 Oracle Rdb and OpenVMS ODS-5 Volumes 5–8
5.1.11 Optimization of Check Constraints 5–8
5.1.12 Carryover Locks and NOWAIT Transaction Clarification 5–10
5.1.13 Unexpected Results Occur During Read-Only Transactions on a Hot Standby Database 5–11
5.1.14 Row Cache Not Allowed While Hot Standby Replication is Active 5–11
5.1.15 Control of Sort Work Memory Allocation 5–12
5.1.16 The Halloween Problem 5–12
5.1.17 Single Statement LOCK TABLE is Not Supported for SQL Module Language and SQL Precompiler 5–14
5.1.18 Multistatement or Stored Procedures May Cause Hangs 5–14
5.1.19 Use of Oracle Rdb from Shareable Images 5–15
5.1.20 RMU/CONVERT Fails When Maximum Relation ID is Exceeded 5–16
5.1.21 RMU/UNLOAD/AFTER_JOURNAL Requires Accurate AIP Logical Area Information 5–16
5.1.22 Do Not Use HYPERSORT with RMU/OPTIMIZE/AFTER_JOURNAL Command 5–17
5.1.23 RMU/BACKUP Operations Should Use Only One Type of Tape Drive 5–18
5.1.24 RMU/VERIFY Reports PGSPAMENT or PGSPMCLST Errors 5–18
5.1.25 Converting Single-File Databases 5–19
5.1.26 Row Caches and Exclusive Access 5–19

5.1.27 Exclusive Access Transactions May Deadlock with RCS Process 5–20
5.1.28 Strict Partitioning May Scan Extra Partitions 5–20
5.1.29 Restriction When Adding Storage Areas with Users Attached to Database 5–21
5.1.30 Multiblock Page Writes May Require Restore Operation 5–21
5.1.31 Replication Option Copy Processes Do Not Process Database Pages Ahead of an Application 5–21
5.1.32 Different Methods of Limiting Returned Rows from Queries 5–22
5.1.33 Suggestions for Optimal Use of SHARED DATA DEFINITION Clause for Parallel Index Creation 5–23
5.1.34 Side Effect When Calling Stored Routines 5–25
5.1.35 Considerations When Using Holdable Cursors 5–26

Tables

3–1 System Privileges 3–16
4–1 Rdb$USAGE Field Values 4–3
4–2 Optional System Tables and Their Storage Map Names 4–7
4–3 Hot Standby Logical Names 4–14
4–4 Columns for Table EPC$1_221_TRANS_TPB 4–23
4–5 Columns for Table EPC$1_221_TRANS_TPB_ST 4–24
4–6 Columns for Table EPC$1_221_DATABASE 4–24
4–7 Columns for Table EPC$1_221_REQUEST_ACTUAL 4–24
4–8 Columns for Table EPC$1_221_TRANSACTION 4–28
4–9 Columns for Table EPC$1_221_REQUEST_BLR 4–32
5–1 Sort Memory Logicals 5–12
5–2 Elapsed Time for Index Creations 5–25

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb Release 7.4.1.0. 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 7.4.1.0.

Access to Oracle Support

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

Document Structure

This manual consists of the following chapters:

Chapter 1	Describes how to install Oracle Rdb Release 7.4.1.0.
Chapter 2	Describes problems corrected in Oracle Rdb Release 7.4.1.0.
Chapter 3	Describes enhancements introduced in Oracle Rdb Release 7.4.1.0.
Chapter 4	Provides information not currently available in the Oracle Rdb documentation set.
Chapter 5	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb Release 7.4.1.0.

1

Installing Oracle Rdb Release 7.4.1.0

This software update is installed using the OpenVMS VMSINSTAL utility.

NOTE

Oracle Rdb Release 7.4 kits are full kits. There is no requirement to install any prior release of Oracle Rdb when installing new Rdb Release 7.4 kits.

1.1 Oracle Rdb on OpenVMS Industry Standard 64

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

The database format for Oracle Rdb Release 7.4 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 7.4 database from prior Rdb versions (on Alpha or VAX platforms) or from other systems on the network is available via the Oracle Rdb remote database server.

1.2 Requirements

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

- This Oracle Rdb release requires the following OpenVMS environments:
  - OpenVMS Alpha version 8.4 or later.
  - OpenVMS Industry Standard 64 version 8.4 or later.
- Some hardware configurations require the installation of OpenVMS V8.4-1H1 or later from VMS Software Inc. (VSI).
- Oracle strongly recommends that all available OpenVMS patches are installed on all systems prior to installing Oracle Rdb. Contact your HPE or VSI support representative for more information and assistance.
- Oracle Rdb must be shutdown before you install this update kit. That is, the command file SYS$STARTUP:RMONSTOP74.COM should be executed before proceeding with this installation. If you have an OpenVMS cluster, you must shutdown the Rdb Release 7.4 monitor on all nodes in the cluster before proceeding.
- After executing RMONSTOP74.COM, no process on any system in the cluster should have any existing RDMSHRP74.EXE image activated. See Section 1.2.1 for additional information.

- The installation requires approximately 280,000 blocks for OpenVMS Alpha systems.
- The installation requires approximately 500,000 blocks for OpenVMS I64 systems.

1.2.1 Ensure No Processes Have RDMSHRP Image Activated

The Oracle Rdb installation procedure checks to make sure that the Oracle Rdb Monitor (RDMMON) process is not running. However, it is also important to make sure that there are no processes on the cluster that share the system disk that have image activated a prior Rdb 7.4 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 RDMSHRP74.EXE image. This procedure should be executed by a privileged account after RMONSTOP74 has been run. Any processes that have RDMSHRP74.EXE activated at this point should be terminated prior to starting the Rdb installation procedure.

```
$ DEFINE /NOLOG /USER RDB$TMP 'RDB$TMP
$ ANALYZE /SYSTEM
    SET OUTPUT RDB$TMP
    SHOW PROCESS /CHANNELS ALL
    EXIT
$ SEARCH /OUTPUT='RDB$TMP' 'RDB$TMP';-1 RDMSHRP74.EXE,"PID:"
$ SEARCH 'RDB$TMP' RDMSHRP74.EXE /WINDOW=(1,0)
$ DELETE /NOLOG 'RDB$TMP';*
```

In the following example, the process 2729F16D named "FOO$SERVER" has the image RDMSHRP74.EXE activated even after RMONSTOP74.COM has been executed and this process is terminated prior to starting the Rdb installation procedure:

```
$ @SYS$STARTUP:RMONSTOP74.COM
   .
   .
   .
$ @FIND_RDMSHRP74_PROC.COM

OpenVMS system analyzer

Process index: 016D   Name: FOO$SERVER  Extended PID: 2729F16D
 0240  7FEF4460 8384F300 $1$DGA2:[VMS$COMMON.SYSLIB]RDMSHRP74.EXE;98

$ STOP/IDENTIFICATION=2729F16D
```

1.3 Deprecated Features

1.3.1 Deprecated Support for the PL/I Compiler

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

1.4 Intel Itanium Processor 9700 “Kittson” Certified

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

1.5 Hardware Emulation Support

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

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

1.6 Maximum OpenVMS Version Check

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

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

1.7 Database Format Changed

The Oracle Rdb on-disk database format is 740 as shown in the following example.

```
$ RMU/DUMP/HEADER databasename
...
  Oracle Rdb structure level is 74.0
...
```

An RMU/CONVERT operation is required for databases created by or accessed by Oracle Rdb V7.0, V7.1, V7.2 or V7.3 to be accessed with Rdb Release 7.4.

Prior to upgrading to Oracle Rdb Release 7.4 and prior to converting an existing database to Oracle Rdb Release 7.4 format, Oracle strongly recommends that you perform a full database verification (with the “RMU /VERIFY /ALL” command) along with a full database backup (with the “RMU /BACKUP” command) to ensure a valid and protected database copy.

1.8 Using Databases from Releases Earlier than V7.0

The RMU Convert command for Oracle Rdb V7.4 supports conversions from Rdb V7.0, V7.1, V7.2 and V7.3 format databases only. This restriction also applies to the RMU Restore command which implicitly uses RMU Convert during the restore operation.

```
$ RMU/RESTORE/NOCDD/DIRECTORY=SYS$DISK:[] [-]TESTING.RBF
%RMU-I-AIJRSTAVL, 0 after-image journals available for use
%RMU-I-AIJISOFF, after-image journaling has been disabled
%RMU-I-LOGCONVRT, database root converted to current structure level
%RMU-S-CVTDBSUC, database DEV1:[DOCUMENTATION.T]TESTING.RDB;1 successfully
converted from version V7.0 to V7.4
%RMU-I-CVTCOMSUC, CONVERT committed for DEV1:[DOCUMENTATION.T]TESTING.RDB;1
to version V7.4
%RMU-W-USERECCOM, Use the RMU Recover command. The journals are not available.
```

If you have an Oracle Rdb V3.0 through V6.1 format database or database backup, you must convert it to at least Oracle Rdb V7.0 format and then convert it to Oracle Rdb V7.4 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 V7.4 format. This might be achieved by using Rdb V6.1 in an OpenVMS Alpha multiversion environment to convert from V4.2 to V6.1. Then installing Rdb V7.1 in that multiversion environment to convert V6.1 to V7.1. Then installing Rdb V7.4 in that multiversion environment to convert V7.1 to V7.4.

Note

This multi-step conversion will require you to use RMU/CONVERT/COMMIT so rolling back to Rdb V4.2 from V7.4 will not be possible. Oracle recommends that full database backups be performed prior to starting conversions to later versions.

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

If you attempt to convert or restore a database format that is prior to Oracle Rdb V7.0 format directly to Oracle Rdb V7.4 format, Oracle RMU generates an error.

```
$ rmu/convert scratch/noconfirm
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.4-10 on OpenVMS Alpha V8.4
%RMU-F-CVRTUNS, The minimum database version that can be converted is version
70.
%RMU-F-FTL_CNV, Fatal error for CONVERT operation at 22-APR-2015 15:48:04.61
```

1.9 Changes to Installation Procedure

Bug 30622247

The Oracle Rdb installation procedure has been enhanced to support mixed case passwords and the VSI Enhanced Password Management software.

These changes affect the installation of Oracle Rdb when it is installed on a system that does not have one or more of the required Rdb accounts. These accounts are RDB$REMOTE74 (support account for remote access), RDMAIJ74 (support account for Hot Standby and the Log Shipping Services) and RDMSTT74 (support account for cluster-wide RMU Show Statistics).

Typically, when upgrading the Rdb installation to a point release, these accounts will already exist and in that case will not be re-created.

The following changes have been made to the Oracle Rdb installation procedures.

- If the installer account has the UAF flag PWDMIX set, then Oracle Rdb no longer forces the generated password (provided by OpenVMS VMSINSTAL.COM) to upper case. This means that the created accounts (RDB$REMOTE74, RDMAIJ74 and RDMSTT74) will also have the UAF flag PWDMIX set and use mixed case passwords.

  If the VSI (VMS Software, Inc.) Enhanced Password Management software is installed, then it is likely that other (numeric, mixed case, and special) characters will also be generated. Oracle Rdb now supports the use of such enhanced passwords. However, if the generated password contains either single (') or double (") quotes, then it will be rejected and Rdb will request an alternate password.

  Note

  A revised version of the OpenVMS VMSINSTAL.COM procedure (located in SYS$UPDATE) is required to compliment the changes made to the Oracle Rdb installation procedure. Please ensure that the VSI OpenVMS UTILITY V1.0 patch kit has been installed prior to installing Oracle Rdb.

- If the installer account does not have the UAF flag PWDMIX set, then the password will be upper case as in previous releases.

Please refer to VMS Software, Inc. for details and availability of the VSI Enhanced Password Management software for VSI OpenVMS Integrity Version 8.4-2L1 and VSI OpenVMS Alpha Versions 8.4-2L2 and 8.4-2L1.

1.10 Invoking the VMSINSTAL Procedure

The installation procedure is the same for Oracle Rdb for OpenVMS Alpha and Oracle Rdb for OpenVMS I64.

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

- To install the Oracle Rdb for OpenVMS I64 kit

```
@SYS$UPDATE:VMSINSTAL RDBV74100IM074 device-name
```

- To install the Oracle Rdb for OpenVMS Alpha kit

```
@SYS$UPDATE:VMSINSTAL RDBV74100AM074 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]`

1.11 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.

1.12 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, "RDBVMSV7.4".

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=RDBVMSV7.4 -
_$ /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.

1.13 VMS$MEM_RESIDENT_USER Rights Identifier Required

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

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

2

Software Errors Fixed in Oracle Rdb Release 7.4.1.0

This chapter describes software errors that are fixed by Oracle Rdb Release 7.4.1.0.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 RDMS-F-BADPAGNUM Exception After Snapshot File Truncated

Bug 11073409

A bugcheck could occur in the following, rare, scenario:

1. A user updates a table but does not commit the transaction.
2. Another user truncates the snapshot file for the area updated by the first user.
3. Another user starts a read only transaction and attempts to read the table updated by the first user. The query fails and a bugcheck dump is produced.

The bugcheck contains a RDMS-F-BADPAGNUM exception, and the area noted in the error is the snapshot area. For example:

```
***** Exception at 0000000081FF7FC0 : RDMSHRP74\PIOFETCH$WITHIN_DB + 00000CB0
%RDMS-F-CANTREADDBS, error reading pages 4:2-2
-RDMS-F-BADPAGNUM, page 2 is out of valid range (1:1) for physical area 4
```

To avoid this problem, make sure there are no active update transactions accessing an area while a snapshot file truncate operation is active.

This error has been corrected in Oracle Rdb Release 7.4.1.0. All active transactions must now end before a snapshot file may be truncated. Once there is no database transaction activity, the snapshot file is truncated and then new transactions are allowed.

Note that attempting to truncate a snapshot file while there are idle transactions in the database will cause all database activity to stop. Care should be taken when using this feature. Ensure that there are no idle transactions before attempting to truncate a snapshot file.

2.1.2 SQL/Services V7.4 IVP Fails With RMU-E-NOSQSCLIENT

Bug 30895210

The Oracle SQL/Services Release 7.4 IVP fails with the following error on a system where Oracle SQL/Services Release 7.3 has never been installed and Oracle Rdb 7.3 or 7.4 is being executed.

```
$ RMU/EXECUTE/COMMAND "RMU/SHOW VERSION"
%RMU-F-ERREXCCMD, Error executing command "RMU/SHOW VERSION".
-RMU-E-NOSQSCLIENT, Cannot find image "SQLSRV_LIBCLIENT"
%RMU-F-FTL_RMU, Fatal error for RMU operation at 12-FEB-2020 11:03:20.19
```

RMU was attempting to activate the Oracle SQL/Services image SYS$LIBRARY:SQLSRV_LIBCLIENT73.EXE, rather than using the logical SQLSRV_LIBCLIENT to activate the desired image. Also, each release's Oracle SQL/Services startup and shutdown procedure was defining the logical SQLSRV_LIBCLIENT to the version specific image. That resulted in the logical being defined for whichever release of Oracle SQL/Services was last started up.

The SQLSRV_LIBCLIENT logical should be defined as a system logical in the system startup procedure, SYS$STARTUP:SYSTARTUP_VMS.COM, to point to the Oracle SQL/Services SQLSRV_LIBCLIENTnn image, where nn is the release of Oracle SQL/Services where the RMU_DISP dispatcher and RMU_SERVICE service are executing. These can only be running for one release of Oracle SQL/Services since they only execute using a predefined port that cannot be shared among different releases.

For example:

```
$ DEFINE/SYSTEM SQLSRV_LIBCLIENT SYS$LIBRARY:SQLSRV_LIBCLIENT74.EXE
```

This problem has been corrected in Oracle Rdb Release 7.4.1.0 and Oracle SQL/Services 7.4. RMU will activate the image defined by the SQLSRV_LIBCLIENT logical and Oracle SQL/Services procedures will no longer define or deassign the logical.

2.1.3 Long Running Delete Does Not Advance After-Image Journal Checkpoint

Bug 6323701

If a database attach executes a data manipulation operation, like a delete, that takes a very long time to complete, it might not move its checkpoint location. In extreme circumstances, this could lead to all after-image journals filling without the possibility of a backup. This kind of problem is often refered to as a "long verb" issue.

To avoid this problem, try to break up large data manipulation operations into smaller ones.

This problem has been corrected in Oracle Rdb Release 7.4.1.0. Oracle Rdb now periodically checks to see if it has items on its work queue waiting to be executed. If there is a waiting request, it will be processed and then the current operation will continue. This should allow checkpoint locations to be advanced as the journals fill.

2.1.4 Long Running Query Blocks Other Queries

Bug 12972389

If a database user executes a long running query and it is holding a lock needed by another user, it might not release that lock for a long period of time, causing other users to wait. This kind of problem is often refered to as a "long verb" issue.

This problem has been corrected in Oracle Rdb Release 7.4.1.0. Oracle Rdb now periodically checks to see if it has items on its work queue waiting to be executed. If there is a waiting request, it will be processed and then the current operation will continue. This should prevent most long running queries from blocking other users.

2.2 SQL Errors Fixed

2.2.1 Unexpected Bugcheck When Using DROP STORAGE AREA ... CASCADE

Bug 30815275

In prior releases of Oracle Rdb, it was possible for ALTER DATABASE to bugcheck if one of the storage areas was missing. Specifically, when using ALTER DATABASE ... DROP STORAGE AREA ... CASCADE for an area that is referenced by an index or a table's storage map, this bugcheck could occur.

This problem has been corrected in Oracle Rdb Release 7.4.1.0. Oracle Rdb has been corrected and this is now the expected output from such a command under these conditions.

```
alter database
    filename mf_personnel
    drop storage area sample_data cascade
;
%RDB-F-SYS_REQUEST, error from system services request
-RDMS-F-FILACCERR, error opening storage area file
DISK:[DIRECTORY]SAMPLE_DATA.RDA;1
-RMS-E-FNF, file not found
```

2.2.2 Unexpected Error %SQL-F-FLDNOTCRS For Declaring a LIST Cursor

Bug 31028255

In previous releases of Oracle Rdb, the DECLARE CURSOR statement for a LIST cursor may generate an unexpected error as shown in the following example:

```
SQL> declare curs_dyn_t cursor for
cont>     select rr.resume res1, r2.resume res2
cont>      from resumes rr, resumes r2
cont>      where rr.employee_id = r2.employee_id;
SQL>
SQL> open curs_dyn_t;
SQL> fetch curs_dyn_t;
 RES1                      RES2
               1:720:1                  1:720:1
SQL>
SQL> declare curs_dyn_l list cursor for
cont>     select res1 where current of curs_dyn_t;
%SQL-F-FLDNOTCRS, Column RESUME was not found in the tables in current scope
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.4.1.0.

2.3 RMU Errors Fixed

2.3.1 Clarification of RMU Unload After_Image Header

Bug 31191945

In prior versions of Oracle Rdb, the LogMiner header returned with each row describes the RDB$LM_DATA_LEN and RDB$LM_RECORD_VERSION as SIGNED WORD (or SMALLINT). However, these fields are, in fact, unsigned 16 bit integers.

An Oracle Rdb table can have a total column length that exceeds 32767, and such values were previously displayed in TEXT, DELIMITED_TEXT and DUMP formats as negative numbers.

The following example shows the definition from the record-definition file.

```
DEFINE FIELD RDB$LM_DATA_LEN DATATYPE IS SIGNED WORD.
DEFINE FIELD RDB$LM_RECORD_VERSION DATATYPE IS SIGNED WORD.
```

The following example shows the generated table definition for SQL. In this case, both the columns RDB$LM_DATA_LEN and RDB$LM_RECORD_VERSION are described as SMALLINT because SQL doesn't have a 16 bit unsigned integer.

```
-- Table definition for LogMiner transaction data 12-MAY-2020 11:47:19.70
-- From database table "SAMPLE2"
CREATE TABLE RDB_LM_SAMPLE2 (
  RDB$LM_ACTION CHAR
 ,RDB$LM_RELATION_NAME CHAR (31)
 ,RDB$LM_RECORD_TYPE INTEGER
 ,RDB$LM_DATA_LEN SMALLINT
 ,RDB$LM_NBV_LEN SMALLINT
 ,RDB$LM_DBK BIGINT
 ,RDB$LM_START_TAD DATE VMS
 ,RDB$LM_COMMIT_TAD DATE VMS
 ,RDB$LM_TSN BIGINT
 ,RDB$LM_RECORD_VERSION SMALLINT
 ,IDENT INTEGER
 ,COMMENT CHAR (1000)
 ,DETAILS CHAR (64250)
);
```

Applications that provide a callable image to receive the extracted table rows should define the header fields as shown below.

```
#include <ints.h>

#pragma member_alignment __save
#pragma nomember_alignment

typedef struct { /* LogMiner structure Header */
    char        rdb$lm_action;
    char        rdb$lm_relation_name [31];
    int         rdb$lm_record_type;
unsigned short  rdb$lm_data_len;
    short       rdb$lm_nbv_len;
    dbk         rdb$lm_dbk;
    int64       rdb$lm_start_tad;
    int64       rdb$lm_commit_tad;
    int64       rdb$lm_tsn;
unsigned short  rdb$lm_record_version;
    } lmrHeader;

#pragma member_alignment __restore
```

In prior versions of Oracle Rdb, the RMU Unload After_Image command would incorrectly process these fields as signed and therefore display values as negative values in the DELIMITED_TEXT, DUMP and TEXT format. This has been corrected in Oracle Rdb Release 7.4.1.0.

2.3.2 Unexpected RMU/UNLOAD/AFTER Bugcheck at AIJEXT$QSORT_RELEASE_REC

Bug 31191945

In prior releases of Oracle Rdb, it was possible for the RMU/UNLOAD/AFTER command to bugcheck when processing rows for a table with long record sizes (greater than 32767 bytes).

This problem has been corrected in Oracle Rdb Release 7.4.1.0.

3

Enhancements And Changes Provided in Oracle Rdb Release 7.4.1.0

3.1 Enhancements And Changes Provided in Oracle Rdb Release 7.4.1.0

3.1.1 New PCSI Support for Rdb Kit Installation and Deinstallation

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

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

```
The following product has been selected:
    ORCL I64VMS RDB74 V7.4-100              Transition (registration)

The following product will be registered:
    ORCL I64VMS RDB74 V7.4-100              DISK$NODE84_2:[VMS$COMMON.]

File lookup pass starting ...

Portion done: 0%
...100%

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

The following product has been registered:
    ORCL I64VMS RDB74 V7.4-100             Transition (registration)
%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target directories...
```

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

```
$ PRODUCT SHOW HISTORY/SINCE
------------------------------------ ----------- ----------- --- -----------
PRODUCT                              KIT TYPE    OPERATION   VAL DATE
------------------------------------ ----------- ----------- --- -----------
ORCL I64VMS RDB74 V7.4-100           Transition  Reg Product (U) 10-JUN-2020
------------------------------------ ----------- ----------- --- -----------

1 item found

$ PRODUCT SHOW HISTORY RDB7*
------------------------------------ ----------- ----------- --- -----------
PRODUCT                              KIT TYPE    OPERATION   VAL DATE
------------------------------------ ----------- ----------- --- -----------
ORCL I64VMS RDB74 V7.4-100           Transition  Reg Product (U) 10-JUN-2020
------------------------------------ ----------- ----------- --- -----------
```

```
1 item found

$ PRODUCT SHOW PRODUCT RDB7*
---------------------------------- ----------- ---------
PRODUCT                            KIT TYPE    STATE
---------------------------------- ----------- ---------
ORCL I64VMS RDB74 V7.4-100         Transition  Installed
---------------------------------- ----------- ---------

1 item found
```

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

```
The following product has been selected:
    ORCL I64VMS RDB74 V7.4-100            Transition (registration)

The following product will be removed from destination:
    ORCL I64VMS RDB74 V7.4-100            DISK$CLYPPR84_2:[VMS$COMMON.]

Portion done: 0%...100%

The following product has been removed:
    ORCL I64VMS RDB74 V7.4-100            Transition (registration)
```

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

```
$ PRODUCT SHOW HISTORY/SINCE
---------------------------------- ----------- ----------- --- -----------
PRODUCT                            KIT TYPE    OPERATION   VAL DATE
---------------------------------- ----------- ----------- --- -----------
ORCL I64VMS RDB74 V7.4-100         Transition  Remove       -  10-JUN-2020
ORCL I64VMS RDB74 V7.4-100         Transition  Reg Product (U) 10-JUN-2020
---------------------------------- ----------- ----------- --- -----------
2 items found

$ PRODUCT SHOW HISTORY RDB7*
---------------------------------- ----------- ----------- --- -----------
PRODUCT                            KIT TYPE    OPERATION   VAL DATE
---------------------------------- ----------- ----------- --- -----------
ORCL I64VMS RDB74 V7.4-100         Transition  Remove       -  10-JUN-2020
ORCL I64VMS RDB74 V7.4-100         Transition  Reg Product (U) 10-JUN-2020
---------------------------------- ----------- ----------- --- -----------
2 items found

$ PRODUCT SHOW PRODUCT RDB7*
---------------------------------- ----------- ---------
PRODUCT                            KIT TYPE    STATE
---------------------------------- ----------- ---------
0 items found
```

3.1.2 Some Aggregate Functions Now Inherit Source Column EDIT STRING

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

- MAX, MEDIAN, MIN, FIRST_VALUE, LAST_VALUE

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

- CAST

When the datatype of the CAST includes a domain with the EDIT STRING.

The following example shows the EDIT STRING being used.

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

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

       $233.00         $100.00         $166.50
1 row selected
SQL>
```

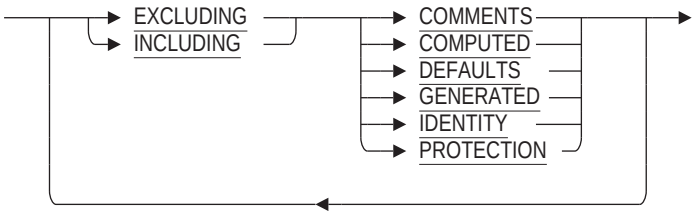
Use the SET DISPLAY NO EDIT STRING statement to disable this behavior.

3.1.3 Enhanced LIKE Table Support in CREATE TABLE Statement

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

Syntax

like-attributes =



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

3.1.4 Comma Statement Separator in Trigger Body No Longer Supported

The syntax for trigger actions in the CREATE TRIGGER statement has, in the past, supported the comma (,) as well as the semicolon (;) as statement separators. The use of the comma separator has been problematic in Oracle Rdb SQL because it conflicts in various places with the comma used as an element separator within some statements. For example, the TRACE statement allows a comma separated list of values and the INSERT INTO ... SELECT ... FROM statement allows a comma separated list of table names in the FROM clause. In these cases, a comma cannot be used as a statement separator because the current statement appears to be continued.

Future versions of Oracle Rdb are expected to include enhancements to the TRIGGER action syntax which will allow other statements to include comma as an element separator. Therefore, the comma statement separator is now no longer supported. This functionality has been deprecated since Rdb V7.2.5.6 and V7.3.2 in the release notes and also reported by the SQL CREATE TRIGGER statement.

Any scripts or applications that include the CREATE TRIGGER statement must now be modified to use only the semicolon (;) as a separator.

This change does not affect existing database triggers, only new triggers defined using the CREATE TRIGGER statement. The RMU Extract Item=TRIGGER command already generates semicolon separators in extracted CREATE TRIGGER statements.

3.1.5 New RMU RECLAIM /FREE_PAGES Qualifier Frees Unused Data Page Clumps

There is a new /FREE_PAGES qualifier for the RMU Reclaim command. This qualifier is used to free unused data page clumps that are allocated in uniform storage areas. It will free all unused page clumps in an entire uniform storage area or all unused page clumps in one or more specified table or index logical areas in uniform storage areas. Any deleted dbkeys and locked space on pages will also be freed.

Command Qualifiers

/[NO]FREE_PAGES

Nofree_pages is the default.

Other qualifiers may be used in conjunction with the Free_pages qualifier.

/AREA[=storage-area-name-list]

Area is used to specify a list of uniform storage area names to process. The wild card syntax AREA=* can be specified for processing all uniform storage areas in the database.

The default for the Area qualifier is all uniform storage areas in the database.

/LAREA=logical-area-name-list

Larea is used to specify a list of individual table or index logical area names to process.

There is no default for the Larea qualifier. A list of logical area names must be specified. The logical area name will be used to determine the storage area where the logical area is located. If the logical area is partitioned among multiple storage areas, each logical area partition will be processed.

This qualifier can only be specified if the Free_pages qualifier is specified.

/LOCK_TIMEOUT=seconds

Lock_timeout is used to specify a lock timeout value that will be in effect during the execution of the RMU/RECLAIM/FREE_PAGES command.

Lock_timeout can only be specified if Free_pages is also specified. The value specified with this qualifier is the maximum time in seconds during which the current RMU/RECLAIM/FREE_PAGES command will wait to acquire an exclusive update lock on the current storage area or logical area to be processed when accessing an on-line database with other users.

If Lock_timeout is not specified, one of the following values will be used, in the specified order of precedence.

1. The value of the logical name RDM$BIND_LOCK_TIMEOUT_INTERVAL, if it has been specified.
2. The "LOCK TIMEOUT INTERVAL" specified by the SQL CREATE or ALTER DATABASE command is used.
3. The RMU/RECLAIM/FREE_PAGES command will wait indefinitely to acquire an exclusive update lock on the current storage area or logical area to be processed.

If /LOCK_TIMEOUT=0 is specified, the RMU/RECLAIM/FREE_PAGES command will ignore any lock timeout defaults that may be in effect and wait indefinitely to acquire an exclusive update lock on the current storage area or logical area to be processed.

Usage Notes

- The Free_pages command can be used when the database is active. Please note that RMU will lock affected areas during processing, which may reduce concurrency.
- Free_pages is not a default qualifier for the RMU Reclaim command. If the Free_pages qualifier is not specified, the RMU Reclaim command will implement the default functionality of freeing deleted dbkeys and locked space in mixed and uniform database storage areas.
- RMU Reclaim Free_Pages can be interrupted at any time; any work in progress will be rolled back. Actions that are completed will have each been committed: if processing a list of logical area names (/LAREA), a commit is performed after each logical area and if processing a list of storage areas (/AREA), a commit is performed after each storage area. Note that tables and indices which are partitioned have multiple logical areas that share the same name as the table or index.
- If Free_pages is specified without either the Larea or Area qualifier, all the uniform storage areas in the database will be processed.
- The Area and Larea qualifiers cannot both be specified in the same RMU Reclaim command.
- If a mixed storage area name is specified with the Area qualifier or the name of a logical area in a mixed storage area is specified with the Larea qualifier, a warning message will be output and a warning status will be returned by the Reclaim command. That storage or logical area will not be processed but

the Reclaim command will continue processing the next storage or logical area in the specified list of storage areas or logical areas.

- If a lock wait timeout occurs, a warning message will be output and a warning status will be returned by the Reclaim command. That storage or logical area will not be processed but the Reclaim command will continue processing the next storage area or logical area in the specified list of storage or logical areas.
- The RMU Reclaim Free_pages functionality replaces that provided by RMU REPAIR /INITIALIZE=FREE_PAGES. The main advantage of Reclaim is that it can be run on an active database.

Examples

Examples using /AREA

The following examples show the Free_pages qualifier with the Area qualifier to free unused page clumps for one or more named storage areas.

```
$ RMU/RECLAIM/LOG/AREA=ABM_AREA1/FREE_PAGES ABM_SAMPLE.RDB
%RMU-I-RCLMAREA, Reclaiming area ABM_AREA1
%RMU-I-RCLMPAGPRC, 2138 pages processed for area ABM_AREA1
%RMU-I-RCLMPAGFREED, 1992 clump pages freed for area ABM_AREA1
$
$ RMU/RECLAIM/FREE_PAGES/AREA=(MFDBA2,MFDBA1)/LOG MFDB
%RMU-I-RCLMAREA, Reclaiming area MFDBA2
%RMU-I-RCLMPAGPRC, 13 pages processed for area MFDBA2
%RMU-I-RCLMPAGFREED, 4 clump pages freed for area MFDBA2
%RMU-I-RCLMAREA, Reclaiming area MFDBA1
%RMU-I-RCLMPAGPRC, 13 pages processed for area MFDBA1
%RMU-I-RCLMPAGFREED, 4 clump pages freed for area MFDBA1
$
$ RMU/RECLAIM/FREE_PAGES/AREA=*/LOG MFDB
%RMU-I-RCLMAREA, Reclaiming area DISK:[DIRECTORY]MFDB.RDA;1
%RMU-I-RCLMPAGPRC, 701 pages processed for area
DISK:[DIRECTORY]MFDB.RDA;1
%RMU-I-RCLMPAGFREED, 220 clump pages freed for area
DISK:[DIRECTORY]MFDB.RDA;1
%RMU-I-RCLMAREA, Reclaiming area DISK:[DIRECTORY]MFDBA1.RDA;1
%RMU-I-RCLMPAGPRC, 13 pages processed for area
DISK:[DIRECTORY]MFDBA1.RDA;1
%RMU-I-RCLMPAGFREED, 0 clump pages freed for area
DISK:[DIRECTORY]MFDBA1.RDA;1
%RMU-I-RCLMAREA, Reclaiming area DISK:[DIRECTORY]MFDBA2.RDA;1
%RMU-I-RCLMPAGPRC, 13 pages processed for area
DISK:[DIRECTORY]MFDBA2.RDA;1
%RMU-I-RCLMPAGFREED, 0 clump pages freed for area
DISK:[DIRECTORY]MFDBA2.RDA;1
%RMU-I-RCLMAREA, Reclaiming area DISK:[DIRECTORY]MFDBA3.RDA;1
%RMU-I-RCLMPAGPRC, 13 pages processed for area
DISK:[DIRECTORY]MFDBA3.RDA;1
%RMU-I-RCLMPAGFREED, 0 clump pages freed for area
DISK:[DIRECTORY]MFDBA3.RDA;1
$
$ RMU/RECLAIM/FREE_PAGES/AREA/LOG MFDB
%RMU-I-RCLMAREA, Reclaiming area DISK:[DIRECTORY]MFDB.RDA;1
%RMU-I-RCLMPAGPRC, 701 pages processed for area
DISK:[DIRECTORY]MFDB.RDA;1
%RMU-I-RCLMPAGFREED, 220 clump pages freed for area
DISK:[DIRECTORY]MFDB.RDA;1
%RMU-I-RCLMAREA, Reclaiming area DISK:[DIRECTORY]MFDBA1.RDA;1
%RMU-I-RCLMPAGPRC, 13 pages processed for area
DISK:[DIRECTORY]MFDBA1.RDA;1
%RMU-I-RCLMPAGFREED, 0 clump pages freed for area
```

```
DISK:[DIRECTORY]MFDBA1.RDA;1
%RMU-I-RCLMAREA, Reclaiming area DISK:[DIRECTORY]MFDBA2.RDA;1
%RMU-I-RCLMPAGPRC, 13 pages processed for area
DISK:[DIRECTORY]MFDBA2.RDA;1
%RMU-I-RCLMPAGFREED, 0 clump pages freed for area
DISK:[DIRECTORY]MFDBA2.RDA;1
%RMU-I-RCLMAREA, Reclaiming area DISK:[DIRECTORY]MFDBA3.RDA;1
%RMU-I-RCLMPAGPRC, 13 pages processed for area
DISK:[DIRECTORY]MFDBA3.RDA;1
%RMU-I-RCLMPAGFREED, 0 clump pages freed for area
DISK:[DIRECTORY]MFDBA3.RDA;1
$
```

Examples using /LAREA

The following examples show the Free_pages qualifier with the Larea qualifier to free unused page clumps for one or more named table and index logical areas.

```
$ RMU/RECLAIM/LOG/LAREA=SAMPLE_TABLE/FREE_PAGES ABM_SAMPLE.RDB
%RMU-I-RCLMLAREA, Reclaiming logical area SAMPLE_TABLE in physical
area DISK:[DIRECTORY]ABM_AREA1.RDA;1
%RMU-I-RCLMLPAGPRC, 2008 pages processed for logical area SAMPLE_TABLE
in physical area DISK:[DIRECTORY]ABM_AREA1.RDA;1
%RMU-I-RCLMLPAGFREED, 1992 clump pages freed for logical area
SAMPLE_TABLE in physical area DISK:[DIRECTORY]ABM_AREA1.RDA;1
$
$ RMU/RECLAIM/LOG/LAREA=(SAMPLE_TABLE,SAMPLE_TABLE2)/FREE_PAGES
 ABM_SAMPLE.RDB
%RMU-I-RCLMLAREA, Reclaiming logical area SAMPLE_TABLE in physical
area DISK:[DIRECTORY]ABM_AREA1.RDA;1
%RMU-I-RCLMLPAGPRC, 2008 pages processed for logical area SAMPLE_TABLE
in physical area DISK:[DIRECTORY]ABM_AREA1.RDA;1
%RMU-I-RCLMLPAGFREED, 1992 clump pages freed for logical area
SAMPLE_TABLE in physical area DISK:[DIRECTORY]ABM_AREA1.RDA;1
%RMU-I-RCLMLAREA, Reclaiming logical area SAMPLE_TABLE2 in physical
area DISK:[DIRECTORY]ABM_AREA2.RDA;1
%RMU-I-RCLMLPAGPRC, 2008 pages processed for logical area
SAMPLE_TABLE2 in physical area DISK:[DIRECTORY]ABM_AREA2.RDA;1
%RMU-I-RCLMLPAGFREED, 1992 clump pages freed for logical area
SAMPLE_TABLE2 in physical area DISK:[DIRECTORY]ABM_AREA2.RDA;1
$
$ RMU/RECLAIM/LOG/LAREA=NDX_NAME/FREE_PAGES TEST_DATABASE.RDB
%RMU-I-RCLMLAREA, Reclaiming logical area NDX_NAME in
physical area DISK:[DIRECTORY]DB_DEFAULT.RDA;1
%RMU-I-RCLMLPAGPRC, 12 pages processed for logical area
NDX_NAME in physical area DISK:[DIRECTORY]DB_DEFAULT.RDA;1
%RMU-I-RCLMLPAGFREED, 3 clump pages freed for logical area
NDX_NAME in physical area DISK:[DIRECTORY]DB_DEFAULT.RDA;1
$
```

Examples using /LAREA and partitioned index logical area

The following example shows the Free_pages qualifier specified in an RMU Reclaim command with the Larea qualifier to free unused page clumps in an index logical area partitioned among different storage areas. Each index logical area partition is processed separately.

```
$ RMU/RECLAIM/FREE_PAGES/LOG/LAREA=INDEXA TEST_DATABASE.RDB
%RMU-I-RCLMLAREA, Reclaiming logical area INDEXA in physical area
DISK:[DIRECTORY]INDEXA_1.RDA;1
%RMU-I-RCLMLPAGPRC, 8 pages processed for logical area INDEXA in
physical area DISK:[DIRECTORY]INDEXA_1.RDA;1
%RMU-I-RCLMLPAGFREED, 3 clump pages freed for logical area INDEXA in
physical area DISK:[DIRECTORY]INDEXA_1.RDA;1
%RMU-I-RCLMLAREA, Reclaiming logical area INDEXA in physical area
DISK:[DIRECTORY]INDEXA_2.RDA;1
%RMU-I-RCLMLPAGPRC, 8 pages processed for logical area INDEXA in
physical area DISK:[DIRECTORY]INDEXA_2.RDA;1
%RMU-I-RCLMLPAGFREED, 3 clump pages freed for logical area INDEXA in
physical area DISK:[DIRECTORY]INDEXA_2.RDA;1
%RMU-I-RCLMLAREA, Reclaiming logical area INDEXA in physical area
DISK:[DIRECTORY]INDEXA_3.RDA;1
%RMU-I-RCLMLPAGPRC, 8 pages processed for logical area INDEXA in
physical area DISK:[DIRECTORY]INDEXA_3.RDA;1
%RMU-I-RCLMLPAGFREED, 3 clump pages freed for logical area INDEXA in
physical area DISK:[DIRECTORY]INDEXA_3.RDA;1
%RMU-I-RCLMLAREA, Reclaiming logical area INDEXA in physical area
DISK:[DIRECTORY]INDEXA_4.RDA;1
%RMU-I-RCLMLPAGPRC, 8 pages processed for logical area INDEXA in
physical area DISK:[DIRECTORY]INDEXA_4.RDA;1
%RMU-I-RCLMLPAGFREED, 3 clump pages freed for logical area INDEXA in
physical area DISK:[DIRECTORY]INDEXA_4.RDA;1
$
```

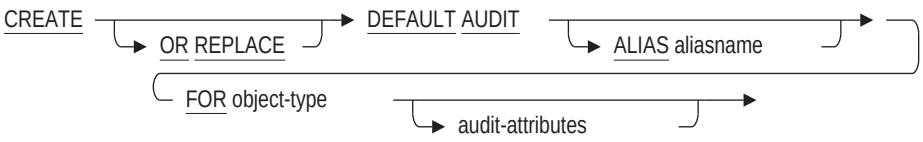
Examples showing warnings

In the following examples, warning messages are output even if /LOG is not specified in the RMU/RECLAIM/FREE_PAGES commands if a mixed storage area is specified or a logical area could not be processed because of a lock conflict with another user. The optional Lock_timeout qualifier is specified.

```
$ RMU/RECLAIM/LOG/AREA=DEPARTMENTS/FREE_PAGES MF_PERSONNEL.RDB
%RMU-W-RCLMMIXIGN, Mixed area DEPARTMENTS not processed if
RMU/RECLAIM/FREE_PAGES
$
$ RMU/RECLAIM/NOLOG/AREA=ABM_AREA1/FREE_PAGES/LOCK_TIMEOUT=600 -
$_ ABM_SAMPLE.RDB
%RMU-W-RCLMARNOTPRC, Area ABM_AREA1 could not be processed due to a
lock conflict
$
$ RMU/RECLAIM/LOG/LAREA=SAMPLE_TABLE/FREE_PAGES/LOCK_TIMEOUT=1200 -
$_  ABM_SAMPLE.RDB
%RMU-I-RCLMLAREA, Reclaiming logical area SAMPLE_TABLE in physical
area DISK:[DIRECTORY]ABM_AREA1.RDA;1
%RMU-W-RCLMLARNOTPRC, Logical area SAMPLE_TABLE could not be processed
due to a lock conflict
%RMU-I-RCLMLPAGPRC, 0 pages processed for logical area SAMPLE_TABLE in
physical area DISK:[DIRECTORY]ABM_AREA1.RDA;1
%RMU-I-RCLMLPAGFREED, 0 clump pages freed for logical area
SAMPLE_TABLE in physical area DISK:[DIRECTORY]ABM_AREA1.RDA;1
$
```

3.1.6 CREATE DEFAULT AUDIT Now Supports CREATE OR REPLACE Syntax and Semantics

This release of Oracle Rdb enhances the CREATE DEFAULT AUDIT statement by allowing the OR REPLACE clause.



Arguments

- OR REPLACE

  If the OR REPLACE clause is used and the referenced object-type exists, then it will be modified using the specified audit flags and comment. Any attributes that are not specified will assume their default values.

  Note

  Any protections granted to the object by the GRANT statement are not replaced. They would need to be removed using the REVOKE statement.

  If the referenced object-type does not exist, then it will be created as if the OR REPLACE clause was not used.

Example

This example shows the CREATE DEFAULT AUDIT statement adding a new table object (which is always named RDB$DEFAULT_AUDIT_TABLE).

```
SQL> create default audit
cont>     for table
cont>     all privileges
cont>     comment is 'Add a default audit table so we can inherit an ACL'
cont> ;
SQL>
SQL> grant select, delete, update, insert, show on rdb$default_audit_table to
testuser2;
SQL> grant select, dbctrl on rdb$default_audit_table to testuser3;
SQL> grant show on rdb$default_audit_table to public;
SQL>
SQL> --> display the attributes for the default table
SQL> --> note: that TESTUSER1 was the executor of the CREATE statement
SQL> show protection on table rdb$default_audit_table;
Protection on Table RDB$DEFAULT_AUDIT_TABLE
    (IDENTIFIER=[TEST,TESTUSER3],ACCESS=SELECT+DBCTRL)
    (IDENTIFIER=[TEST,TESTUSER2],ACCESS=SELECT+INSERT+UPDATE+DELETE+SHOW)
    (IDENTIFIER=[TEST,TESTUSER1],ACCESS=SELECT+INSERT+UPDATE+DELETE+SHOW+
 CREATE+ALTER+DROP+DBCTRL+REFERENCES)
    (IDENTIFIER=[*,*],ACCESS=SHOW)
SQL> show audit on table rdb$default_audit_table;
Audit information for Table RDB$DEFAULT_AUDIT_TABLE
 Audit Privileges:
    ALL
 Alarm Privileges:
    ALL

SQL> show table (comment) rdb$default_audit_table;
Information for table RDB$DEFAULT_AUDIT_TABLE
```

```
Comment on table RDB$DEFAULT_AUDIT_TABLE:
Add a default audit table so we can inherit an ACL

A global temporary table.
On commit Delete rows

SQL>
```

At some later time, the table template object can be created or replaced using the CREATE OR REPLACE DEFAULT AUDIT statement.

```
SQL> create or replace default audit
cont>     for table
cont>     type is (audit)
cont>     privileges (success, failure)
cont>     comment is 'Only audit SUCCESS and FAILURE'
cont> ;
SQL>
SQL> --> show that the protections are retained by OR REPLACE
SQL> show protection on table rdb$default_audit_table;
Protection on Table RDB$DEFAULT_AUDIT_TABLE
    (IDENTIFIER=[TEST,TESTUSER3],ACCESS=SELECT+DBCTRL)
    (IDENTIFIER=[TEST,TESTUSER2],ACCESS=SELECT+INSERT+UPDATE+DELETE+SHOW)
    (IDENTIFIER=[TEST,TESTUSER1],ACCESS=SELECT+INSERT+UPDATE+DELETE+SHOW
 +CREATE+ALTER+DROP+DBCTRL+REFERENCES)
    (IDENTIFIER=[*,*],ACCESS=SHOW)
SQL> show audit on table rdb$default_audit_table;
Audit information for Table RDB$DEFAULT_AUDIT_TABLE
 Audit Privileges:
    SUCCESS,FAILURE

SQL> show table (comment) rdb$default_audit_table;
Information for table RDB$DEFAULT_AUDIT_TABLE

Comment on table RDB$DEFAULT_AUDIT_TABLE:
Only audit SUCCESS and FAILURE

A global temporary table.
On commit Delete rows

SQL>
SQL> commit;
SQL>
```

3.1.7 New System Privileges Feature

This release of Oracle Rdb introduces Database System Privileges - an enhancement for database security.

Introduction

A typical Oracle Rdb database will have a security policy defined by granting privileges to users of the database and, when objects are created (tables, sequences, and so on), granting access to those objects. In addition, roles (also known as rights identifiers) may be granted specific access and any user assigned that role may inherit its access rights.

These granted privileges are stored in an access control list (ACL) with an entry for a user or role known as an access control entry (ACE).

Please refer to the SQL Reference Manual GRANT statement and REVOKE statement sections for more detailed descriptions and examples.

Security Definitions

All objects are within the DATABASE security domain.

The primary security objects are: ASSERTION (an assertion is a form of constraint defined independently of a table definition. This feature is currently not available in SQL but maps to the standalone RDO constraints), CATALOG, COLLATING SEQUENCE, DOMAIN, OUTLINE, PROCEDURE, PROFILE, TABLE, ROLE, SCHEMA, SEQUENCE, SYNONYM, and USER.

For the purposes of system privileges, Oracle Rdb treats modules, functions and procedures as a single class - namely PROCEDURE.

Some primary security objects have an associated ACL that protects that object and controls access to sub-objects. TABLE includes the following sub-objects: VIEW, STORAGE MAP, and TRIGGER.

When TABLE, SEQUENCE, and PROCEDURE objects are created, they are implicitly given an ACL that grants ALL PRIVILEGES to the creator and NO PRIVILEGES to the PUBLIC (also known as [*,*]). The database administrator can override this default ACL for new primary objects:

1. For tables and views, the PUBLIC access control entry will be inherited from the database access control entry for the DEFAULT user. Note: not all OpenVMS systems have a DEFAULT user defined in the system user authorization file (UAF) so that would be created by the system manager if required.

```
SQL> grant show
cont>     on database alias paysys
cont>     to default
cont> ;
SQL>
   .
   .
   .
SQL> create table paysys.CONTROL_TABLE
cont>     (identifier_value integer generated by default as identity
cont>     );
SQL>
SQL> show protection on table paysys.CONTROL_TABLE;
Protection on Table PAYSYS.CONTROL_TABLE
    (IDENTIFIER=[ADMIN,DATABASE],ACCESS=SELECT+INSERT+UPDATE+DELETE+
     SHOW+CREATE+ALTER+DROP+DBCTRL+REFERENCES)
    (IDENTIFIER=[*,*],ACCESS=SHOW)
SQL>
```

2. Alternately, the CREATE DEFAULT AUDIT statement can be used to define template security objects in the database. These special objects are used for audit and protection inheritance. When an object (table, sequence, etc) is created, then the access control list from the security template is inherited. Note that choice (1) above will also be applied to the TABLE and VIEW default audit templates.

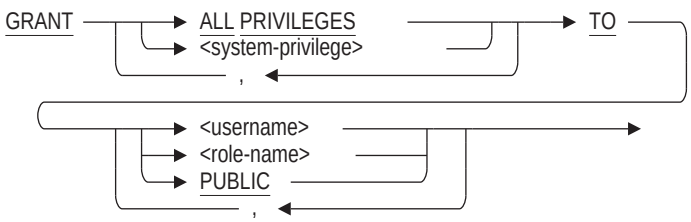
   This example shows the **default audit** object creation and then being granted a default access for PUBLIC and specific access for users and roles.

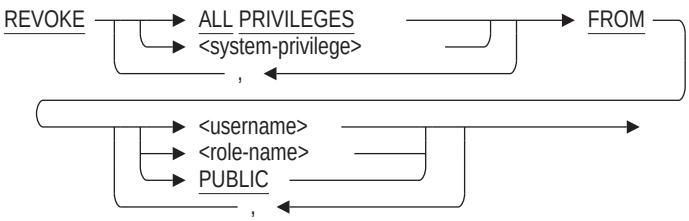
```
SQL> create default audit
cont>     alias PAYSYS
cont>     for table
cont> ;
SQL>
SQL> grant select
cont>     on table PAYSYS.rdb$default_audit_table
cont>     to m_smith, b_lee, s_jain
cont> ;
SQL>
SQL> grant select, insert, delete, update
cont>     on table PAYSYS.rdb$default_audit_table
cont>     to paysys_admin
cont> ;
SQL>
SQL> grant show
cont>     on table PAYSYS.rdb$default_audit_table
cont>     to PUBLIC
cont> ;
SQL>
   .
   .
   .
SQL> create table paysys.CONTROL_TABLE
cont>     (identifier_value integer generated by default as identity
cont>     );
SQL>
SQL> show protection on table paysys.CONTROL_TABLE;
Protection on Table PAYSYS.CONTROL_TABLE
    (IDENTIFIER=PAYSYS_ADMIN,ACCESS=SELECT+INSERT+UPDATE+DELETE)
    (IDENTIFIER=[RDB,S_JAIN],ACCESS=SELECT)
    (IDENTIFIER=[DEV,B_LEE],ACCESS=SELECT)
    (IDENTIFIER=[AUDITOR,M_SMITH],ACCESS=SELECT)
    (IDENTIFIER=[ADMIN,DATABASE],ACCESS=SELECT+INSERT+UPDATE+DELETE+
     SHOW+CREATE+ALTER+DROP+DBCTRL+REFERENCES)
    (IDENTIFIER=[*,*],ACCESS=SHOW)
SQL>
```

Database Vault Feature

The security policy implemented through ACLs can be overridden at runtime by a suitably privileged OpenVMS user. That is, an OpenVMS power user might be able to attach and select data from a table even if they do not have database or table access granted by an access control entry. This override ability can be limited on a per database level by enabling the DATABASE VAULT attribute using ALTER DATABASE ... DATABASE VAULT IS ENABLED. When DATABASE VAULT is enabled, only the access control lists and database system privileges are used to determine access to database objects. This can prevent accidental override of the security policy. See the SQL Reference Manual DATABASE VAULT Appendix for further details.

Syntax





Arguments

- ALL PRIVILEGES

  All privileges will be granted (or revoked) from the listed users and roles.

- system-privilege

  Refer to Table 3–1, System Privileges for a list of supported system privileges.

- TO username
  TO role-name
  TO PUBLIC

  Specifies the user name, role name, or the PUBLIC user to which you want to grant the system privilege. The PUBLIC user is the user name associated with all anonymous users who access the database.

  Note

  Oracle recommends that you only grant system privileges to trusted users. If system privileges are granted to roles then only assign those roles to trusted users.

  If the database is defined as SECURITY CHECK IS INTERNAL and the user or role name exists as an operating system user or rights identifier, Oracle Rdb will automatically create the user or role name when you issue the GRANT statement.

- FROM username
  FROM role-name
  FROM PUBLIC

  Specifies the user, role, or the PUBLIC user from which the specified role is to be revoked.

Database System Privileges

System privileges are associated with specific users (CREATE USER) and roles (CREATE ROLE) within the database. They are assigned by a user with SECURITY privilege on the database by the GRANT statement and removed by the REVOKE statement.

For example,

```
SQL> create user J_JONES identified externally;
SQL> create role DB_PROGRAMMER identified externally;
SQL>
SQL> grant create any sequence, create any procedure,
cont> create any temporary table to DB_PROGRAMMER;
SQL> grant create any trigger, create any index to J_JONES;
```

In this example, the database uses the default SECURITY CHECKING IS EXTERNAL therefore the granting of the rights identifier DB_PROGRAMMER must be performed by OpenVMS. Use the AUTHORIZE utility to grant the rights identifier to specific users.

```
$ run sys$system:authorize
UAF> GRANT/ID DB_PROGRAMMER J_JONES
UAF>
```

When J_JONES attaches to the database, they will receive the benefits of their own granted system privileges (if any) as well as those granted to the assigned role (DB_PROGRAMMER). This is true even for DATABASE VAULT protected databases.

If a database is defined as SECURITY CHECKING IS INTERNAL, then the GRANT statement is used to associate roles created by CREATE ROLE with specific users.

```
SQL> grant DB_PROGRAMMER to J_JONES;
SQL>
```

Refer to the SQL Reference Manual GRANT Statement: Roles section for more details and examples.

Implied System Privileges

In prior releases of Oracle Rdb, there were implied system privileges based on the database level CREATE, ALTER, DROP and SECURITY privileges that may have been granted to the user via the database ACL.

CREATE implies the permission to CREATE ANY object (except PROFILE, ROLE, and USER), and ALTER implies the permission to ALTER any object (except PROFILE, ROLE, and USER), DROP implies the permission to DROP ANY object (except PROFILE, ROLE, and USER), and SECURITY implies the permission to CREATE, ALTER, and DROP ANY PROFILE, ROLE or USER.

These implied system privileges can be displayed when attaching to a database and performing a SHOW PRIVILEGE ON DATABASE command. The implied system privileges (if any) are displayed. This is similar to the access privileges shown by this command. They reflect the ACL on the database as well as inherited access based on special privileges such as DBADM and OpenVMS privileges.

```
SQL> show privileges on database rdb$dbhandle;
Privileges on Alias RDB$DBHANDLE
    (IDENTIFIER=[RDB,RDBUSER2],ACCESS=SELECT+CREATE+ALTER+DROP)

Current system privileges:
     Granted Create Any
     COLLATING SEQUENCE, DOMAIN, ASSERTION, SESSION, OUTLINE, PROCEDURE,
     TABLE, SEQUENCE, VIEW
     Granted Alter Any
     COLLATING SEQUENCE, DOMAIN, ASSERTION, DATABASE, OUTLINE
     Granted Drop Any
     COLLATING SEQUENCE, DOMAIN, ASSERTION, DATABASE, OUTLINE
SQL>
```

Note that ALTER and DROP for primary database objects are not implicitly inherited from the database ALTER and DROP database privileges. This is because these operations are controlled by the object's own access control list.

Fine Tuning

Although system privileges are now available in Oracle Rdb, the database administrator is not required to use them. The database will operate as it did in previous releases.

Making use of this enhanced privilege system requires that the database administrator perform these tasks.

- Use CREATE USER for each database user to which system privileges need to be granted.
- Use CREATE ROLE for any rights identifiers used to fine tune access.

Not all users and roles that access the database need to be created in the database as Rdb will still use them as in prior releases to select matching access control entries. However, users and roles must be created to allow the database administrator to grant system privileges as these objects are used to store the current privilege set.

- Use the GRANT statement to manage the CREATE ANY, ALTER ANY, DROP ANY, and TRUNCATE ANY privileges and assign them to users and roles.
- Use REVOKE on the DATABASE ALIAS to remove the CREATE, ALTER, DROP or SECURITY privileges that were previously granted to those uses. This step is required so that those database level privileges do not interfere with the fine control of system privileges.

This change will limit those users according to the new security policy. The SHOW USER and SHOW ROLE statements will display all the granted system privileges. The SHOW PRIVILEGES ON DATABASE will show the augmented system privileges based on:

- Database ACL entries,
- Granted system privileges for the attaching user,
- Granted system privileges for all rights (roles) assigned to the user,
- OpenVMS privileges (when DATABASE VAULT is enabled there will be none used)

All access control is established at ATTACH time so changes made to the USER, ROLE or DATABASE access will not have an effect until the next database ATTACH.

Note

Oracle recommends that you only grant system privileges to trusted users. If system privileges are granted to roles then only assign those roles to trusted users.

Table 3–1 System Privileges

Operation Type	Object Type	Description
ALL PRIVILEGES		Can be used to GRANT or REVOKE all system privileges to a USER or ROLE.
ALTER ANY ...		
	ASSERTION	Permits the holder to execute the RDO CHANGE CONSTRAINT command and the SQL ALTER CONSTRAINT and COMMENT ON CONSTRAINT statements.
	CATALOG	Permits the holder to execute the ALTER CATALOG statement. The database must have multischema enabled; ALTER DATABASE ... MULTISCHEMA IS ON;
	COLLATING SEQUENCE	Permits the holder to execute the ALTER COLLATING SEQUENCE statement.
	DOMAIN	Permits the holder to execute the ALTER DOMAIN statement.
	DATABASE	Permits the holder to execute the ALTER DATABASE and COMMENT ON DATABASE statements.
	INDEX	Permits the holder to execute the ALTER INDEX statement.
	OUTLINE	Permits the holder to execute the ALTER OUTLINE statement.
	PROCEDURE	Permits the holder to execute the ALTER FUNCTION, ALTER MODULE and ALTER PROCEDURE statements.
	PROFILE	Permits the holder to execute the ALTER PROFILE and ALTER DEFAULT PROFILE statements. When the profile exists, the holder can also execute the CREATE OR REPLACE PROFILE and CREATE OR REPLACE DEFAULT PROFILE statements.
	ROLE	Permits the holder to execute the ALTER ROLE statement.
	SEQUENCE	Permits the holder to execute the ALTER SEQUENCE statement. When the sequence exists, the holder can also execute the CREATE OR REPLACE SEQUENCE statement.
	SCHEMA	Permits the holder to execute the ALTER SCHEMA statement. The database must have multischema enabled: ALTER DATABASE ... MULTISCHEMA IS ON;
	STORAGE MAP	Permits the holder to execute the ALTER STORAGE MAP statement.

(continued on next page)

Table 3–1 (Cont.) System Privileges

Operation Type	Object Type	Description
	SYNONYM	Permits the holder to execute the ALTER SYNONYM statement. When the synonym exists, the holder can also execute the CREATE OR REPLACE SYNONYM statements. The database must have synonyms enabled: ALTER DATABASE ... SYNONYMS ARE ENABLED;
	TABLE	Permits the holder to execute the ALTER TABLE statement.
	TEMPORARY TABLE	Permits the holder to execute the ALTER TEMPORARY TABLE statement or CREATE INFORMATION TABLE statement when CREATE ANY TABLE privilege is not granted.
	TRIGGER	Permits the holder to execute the ALTER TRIGGER statement.
	USER	Permits the holder to execute the ALTER USER statement.
	VIEW	Permits the holder to execute the ALTER VIEW statement. When the view exists, the holder can also execute the CREATE OR REPLACE VIEW statement.
CREATE ...		
	SESSION	Permits the holder to execute ATTACH, CONNECT, DECLARE ALIAS, SET SESSION AUTHORIZATION, and other session starting statements.
CREATE ANY ...		
	ASSERTION	Permits the holder to execute the RDO DEFINE CONSTRAINT command. The SQL equivalent to DEFINE CONSTRAINT would be a table level constraint. Such definitions are managed by TABLE privileges, therefore this privilege does not apply to SQL.
	CATALOG	Permits the holder to execute the CREATE CATALOG statement. The database must have multischema enabled: ALTER DATABASE ... MULTISCHEMA IS ON;
	COLLATING SEQUENCE	Permits the holder to execute the CREATE COLLATING SEQUENCE statement.
	DOMAIN	Permits the holder to execute the CREATE DOMAIN statement.
	INDEX	Permits the holder to execute the CREATE INDEX statement.
	OUTLINE	Permits the holder to execute the CREATE OUTLINE statement.

(continued on next page)

Table 3–1 (Cont.) System Privileges

Operation Type	Object Type	Description
	PROCEDURE	Permits the holder to execute the CREATE FUNCTION, CREATE MODULE and CREATE PROCEDURE statements.
	PROFILE	Permits the holder to execute the CREATE PROFILE and CREATE DEFAULT PROFILE statements.
	ROLE	Permits the holder to execute the CREATE ROLE statement.
	SEQUENCE	Permits the holder to execute the CREATE SEQUENCE statement.
	SCHEMA	Permits the holder to execute the CREATE SCHEMA statement. The database must have multischema enabled: ALTER DATABASE ... MULTISCHEMA IS ON;
	STORAGE MAP	Permits the holder to execute the CREATE STORAGE MAP statement.
	SYNONYM	Permits the holder to execute the CREATE SYNONYM statement. The database must have synonyms enabled: ALTER DATABASE ... SYNONYMS ARE ENABLED;
	TABLE	Permits the holder to execute the CREATE TABLE statement.
	TEMPORARY TABLE	Permits the holder to execute the CREATE TEMPORARY TABLE and CREATE INFORMATION TABLE statements when CREATE ANY TABLE privilege is not granted.
	TRIGGER	Permits the holder to execute the CREATE TRIGGER statement.
	USER	Permits the holder to execute the CREATE USER statement.
	VIEW	Permits the holder to execute the CREATE VIEW statement.
DROP ANY ...		
	ASSERTION	Permits the holder to execute the RDO DELETE CONSTRAINT command or the SQL DROP CONSTRAINT statement.
	CATALOG	Permits the holder to execute the DROP CATALOG statement. The database must have multischema enabled: ALTER DATABASE ... MULTISCHEMA IS ON;
	COLLATING SEQUENCE	Permits the holder to execute the DROP COLLATING SEQUENCE statement.
	DOMAIN	Permits the holder to execute the DROP DOMAIN statement.
	DATABASE	Permits the holder to execute the DROP DATABASE statement.

(continued on next page)

Table 3–1 (Cont.) System Privileges

Operation Type	Object Type	Description
	INDEX	Permits the holder to execute the DROP INDEX statement.
	OUTLINE	Permits the holder to execute the DROP OUTLINE statement.
	PROCEDURE	Permits the holder to execute the DROP FUNCTION, DROP MODULE and DROP PROCEDURE statements.
	PROFILE	Permits the holder to execute the DROP PROFILE and DROP DEFAULT PROFILE statements.
	ROLE	Permits the holder to execute the DROP ROLE statement.
	SEQUENCE	Permits the holder to execute the DROP SEQUENCE statement.
	SCHEMA	Permits the holder to execute the DROP SCHEMA statement. The database must have multischema enabled: ALTER DATABASE ... MULTISCHEMA IS ON;
	STORAGE MAP	Permits the holder to execute the DROP STORAGE MAP statement.
	SYNONYM	Permits the holder to execute the DROP SYNONYM statement. The database must have synonyms enabled: ALTER DATABASE ... SYNONYMS ARE ENABLED;
	TABLE	Permits the holder to execute the DROP TABLE statement.
	TEMPORARY TABLE	Permits the holder to execute the DROP TEMPORARY TABLE statement or DROP INFORMATION TABLE statement when DROP ANY TABLE privilege is not granted.
	TRIGGER	Permits the holder to execute the DROP TRIGGER statement.
	USER	Permits the holder to execute the DROP USER statement.
	VIEW	Permits the holder to execute the DROP VIEW statement.
TRUNCATE ...		
	ANY TABLE	Permits the holder to execute the TRUNCATE TABLE statement. This privilege effectively allows the user to temporarily disable triggers during the TRUNCATE operation and assume DELETE access to the table.
	TABLE	This privilege is similar to the TRUNCATE ANY TABLE privilege but requires that the user also be granted DELETE access to the table. Permits the user to execute the TRUNCATE TABLE statement without further privilege checking for BEFORE and AFTER DELETE triggers.

Usage Notes

- You must have the SECURITY privilege on the database to grant a system privilege to a user or a role.
- You must have the SECURITY privilege on the database to revoke a system privilege from a user or a role.
- The TEMPORARY TABLE class of privileges is considered a subset of TABLE and allows the database administrator to grant privileges to a user but only for logical tables not physical (base) tables.
- If the user has CREATE ANY TABLE, then the CREATE ANY TEMPORARY TABLE privilege is not required. Similarly, ALTER ANY TEMPORARY TABLE and DROP ANY TEMPORARY TABLE are not used if ALTER ANY TABLE or DROP ANY TABLE is granted.
- The SHOW PRIVILEGES ON DATABASE statement displays the current active system privileges. This is based on the current user system privileges, system privileges inherited from granted roles, inherited privileges from the database access control list and OpenVMS process privileges.

  It does not display any privileges for features which are not enabled for the database. CATALOG and SCHEMA privileges will not be displayed if MULTISCHEMA is not enabled. SYNONYM privileges will not be displayed if SYNONYMS are not enabled. However, the SHOW USER and SHOW ROLE statements will display all granted privileges even if that privilege has no application in the current database configuration.

3.1.8 New Database Vault Feature

This release of Oracle Rdb introduces the DATABASE VAULT functionality.

The goal of DATABASE VAULT is to avoid accidental database access by an OpenVMS privileged user when the database security policy (ACL) should prevent such access.

This feature allows the database administrator to enforce an access policy for all attached database users by disabling the use of OpenVMS privileges as overrides to the database access control list.

DATABASE VAULT can be enabled by any of these commands.

- The SQL CREATE DATABASE ... DATABASE VAULT IS ENABLED statement.
- The SQL ALTER DATABASE ... DATABASE VAULT IS ENABLED statement.
- The SQL IMPORT DATABASE ... DATABASE VAULT IS ENABLED statement.

Note

If a database with DATABASE VAULT enabled is exported, then an IMPORT DATABASE will implicitly execute the DATABASE VAULT IS ENABLED action without that clause being required on the statement.

- RMU/SET DATABASE/DATABASE_VAULT=ENABLED command

DATABASE VAULT can be disabled by any of these commands.

- The SQL ALTER DATABASE ... DATABASE VAULT IS DISABLED statement.
- The SQL IMPORT DATABASE ... DATABASE VAULT IS DISABLED statement.
- RMU/SET DATABASE/DATABASE_VAULT=DISABLED command

Before executing any of these commands, the user must be granted (at least temporarily) the rights identifier RDBVMS$DATABASE_VAULT_MANAGER that is added to the system during installation.

For example, the SET RIGHTS_LIST DCL command can be used to temporarily enable it.

```
$ SET RIGHTS_LIST /ENABLE RDBVMS$DATABASE_VAULT_MANAGER
$
$ rmu/set database mf_personnel/database_vault=enable
%RMU-I-MODIFIED, Database state modified
%RMU-W-DOFULLBCK, full database backup should be done to ensure future
recovery
$
$ SET RIGHTS_LIST /DISABLE RDBVMS$DATABASE_VAULT_MANAGER
```

Please refer to the Oracle Rdb SQL Reference Manual, Appendix J for more details. This includes a description of the new DBVAULT audit class that can be used to audit changes to the DATABASE VAULT settings of a database.

3.1.9 Support for VSI TCP/IP for OpenVMS Version 10.6 or Later

Oracle Rdb 7.4 supports VSI TCP/IP for OpenVMS Version 10.6 or later.

The Oracle Rdb installation procedure provides two new command procedures to assist during the installation and for use later by customers for additional service creation. See SYS$LIBRARY for these new procedures: RDB$TCPIP_CREATE_SERVICE.COM and RDB$VSI_TCPIP_RESTART_SERVER.COM.

The Oracle Rdb Installation Guide has been updated to reflect this support and, where necessary, showing examples for both TCP/IP products.

While running the installation, Rdb will detect which TCP/IP software is active and create the RDBSERVER service automatically. The Installation Guide gives instructions for creating your own services using the provided procedures.

RDB$TCPIP_CREATE_SERVICE.COM

This procedure is located in SYS$LIBRARY after the installation of Oracle Rdb V7.4. It can be used to create a TCPIP service for Oracle Rdb for either HPE TCP/IP or VSI TCP/IP.

For example:

```
$ @SYS$LIBRARY:RDB$TCPIP_CREATE_SERVICE -
    RDBSERVER74 -
    9521 RDB$REMOTE74 SYS$COMMON:[SYSEXE]RDBSERVER_TCPIP.COM RDB74 "B" 64
$
```

Parameters:

- P1 = Specifies the name of the service to be created.
- P2 = Specifies the port number to be associated with the service.
- P3 = Specifies the username to be associated with the service.

- P4 = Specifies the command file to be executed for the service.
- P5 = Specifies the process name to be associated with the service. Used by HPE TCPIP only.
- P6 = A flag that specifies what action should be taken:
  - "C" specifies that a service should be created.
  - "R" specifies that the server should be restarted.
  - "B" specifies that both operations should be done (service created and server restarted).
- P7 (optional) = Specifies the limit (maximum servers) to be associated with the service. If not passed, the default is 10.
- P8 (not used) = Reserved for future releases.

RDB$VSI_TCPIP_RESTART_SERVER.COM

This procedure is located in SYS$LIBRARY after the installation of Oracle Rdb V7.4. It can be used to restart the VSI TCPIP server on the current cluster node.

Parameters:

- There are no parameters used by this procedure.

3.1.10 New SET FLAGS Keyword for Hash Join Feature - HASHING

This release of Oracle Rdb has added a new flag to control the HASH JOIN feature of the Rdb optimizer. The HASHING flag can be used with the SET FLAGS statement or the RDMS$SET_FLAGS logical name to enable this feature. When enabled, the optimizer will attempt to engage the HASH JOIN feature during query solution. The default is NOHASHING(JOINS).

The following example shows the HASHING flag in use.

```
SQL> set flags 'strategy,detail(2)';
SQL> set flags 'hashing(joins)';
SQL>
SQL> select e.employee_id, e.birthday, jh.job_start
cont> from employees e, job_history jh
cont> where e.employee_id = jh.employee_id
cont>  and jh.job_end is null
cont> ;
Tables:
  0 = EMPLOYEES
  1 = JOB_HISTORY
Conjunct: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
Hash Q1
  Outer Build
  Match_Key:0.EMPLOYEE_ID
    Get     Retrieval by index of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Inner Probe
  Match_Key:1.EMPLOYEE_ID
    Conjunct: MISSING (1.JOB_END)
    Get     Retrieval by index of relation 1:JOB_HISTORY
      Index name  JH_EMPLOYEE_ID [0:0]
Table=0:EMPLOYEES #Buckets=131 #Hits=61 #Collisions=39 #Dups=0 #Dups_Chain=0
Load_Factor=   4.656488549618321E-001
 E.EMPLOYEE_ID   E.BIRTHDAY    JH.JOB_START
 00164           28-Mar-1947   21-Sep-1981
 00165           15-May-1954    8-Mar-1981
.
.
```

```
.
100 rows selected
SQL>
```

To disable the flag, use 'NOHASHING(JOINS)'. The setting is displayed by the SHOW FLAGS statement.

See also the new logical name RDMS$ENABLE_HASH_JOIN. Defining this logical name to true ("T", "t", "Y", "y" or "1") instructs the Rdb optimizer to try to use in-memory HASH JOIN to solve queries.

3.1.11 New JOIN BY HASH Clause in CREATE OUTLINE Statement

This release of Oracle Rdb adds a new JOIN BY HASH clause to the CREATE OUTLINE statement and the OPTIMIZE OUTLINE clause of the select statement.

The following example shows the new syntax and the resulting query strategy.

```
SQL> create outline QO_1
cont> id '352E2736F133A6A322A3C935DB2CBE12'
cont> mode 0
cont> as (
cont>   query (
cont>     subquery (
cont>       EMPLOYEES 0 access path index EMP_EMPLOYEE_ID
cont>         join by hash to
cont>       SALARY_HISTORY 1 access path index SH_EMPLOYEE_ID
cont>       )
cont>     )
cont>   )
cont> compliance optional;
SQL>
SQL> set flags 'strategy,detail(2)';
SQL>
SQL> select
cont>     e.employee_id, sh.salary_start, sh.salary_amount
cont> from
cont>     employees e
cont>     inner join
cont>     salary_history sh on (e.employee_id = sh.employee_id
cont>                           and sh.salary_end is null)
cont> optimize using QO_1
cont> ;
~S: Outline "QO_1" used
Tables:
  0 = EMPLOYEES
  1 = SALARY_HISTORY
Conjunct: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
Hash Q1
  Outer Build
  Match_Key:0.EMPLOYEE_ID
    Index only retrieval of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Inner Probe
  Match_Key:1.EMPLOYEE_ID
    Conjunct: MISSING (1.SALARY_END)
    Get     Retrieval by index of relation 1:SALARY_HISTORY
      Index name  SH_EMPLOYEE_ID [0:0]
Table=0:EMPLOYEES #Buckets=131 #Hits=61 #Collisions=39 #Dups=0 #Dups_Chain=0
Load_Factor=   4.656488549618321E-001
 E.EMPLOYEE_ID   SH.SALARY_START   SH.SALARY_AMOUNT
 00164           14-Jan-1983             $51,712.00
 00165            1-Jul-1982             $11,676.00
.
.
```

.

Note

This syntax, JOIN BY HASH, cannot be applied remotely to an older version of Oracle Rdb. An error such as this will be returned.

```
SQL> create outline QO_1
cont> id '352E2736F133A6A322A3C935DB2CBE12'
cont> mode 0
cont> as (
cont>   query (
cont>     subquery (
cont>       EMPLOYEES 0 access path index EMP_EMPLOYEE_ID
cont>         join by hash to
cont>       SALARY_HISTORY 1 access path index SH_EMPLOYEE_ID
cont>       )
cont>     )
cont>   )
cont> compliance optional;
%SQL-F-UNSUPVER, Operation is unsupported for version of database
-SQL-F-UNSUPFEATURE, feature JOIN BY HASH is not supported
```

3.1.12 New Hash Join Feature

Status: BETA

This release of Oracle Rdb includes a new optimization method known as Hash Join. A Hash Join is performed by hashing (mapping) one set of data into virtual memory based on the join columns and reading the other table to probe into the hash table to locate matching rows.

Typically, a Hash Join has a lower cost compared to the alternate of sorting when the hash table can be held entirely in memory, with the total cost amounting to very little other than the cost of reading the data sets. The cost rises if the hash table has to be spilled over to a temporary file.

Hash Join is only used for equi-joins. In general, Hash Join is a better solution for joining large numbers of rows in an equi-join.

Applications that in the past used Match Join might (unknowingly) rely on the implicit use of SORT during the query solution. However, as no implicit SORT is performed, the data might appear in a different order with Hash Join. Adding an ORDER BY will sort the result data but no longer sort the inputs to the join.

Note

Not all queries will be solved using HASH JOIN if this optional feature is enabled. Use the `SET FLAGS 'STRATEGY,DETAIL(3)'` to see a report of the current restrictions which cause the HASH JOIN method to be rejected.

Enabling HASH JOIN

The optimizer, by default, does not try the HASH JOIN method in solutions. This action must be enabled by the programmer in one of the following ways. The optimizer will then include HASH JOIN as part of its solution matching, but it may be rejected for various reasons.

- Define the logical name RDMS$ENABLE_HASH_JOIN.

Defining this logical name to true ("T", "t", "Y", "y" or "1") instructs the Rdb optimizer to try to use in-memory HASH JOIN to solve queries.

- Defining the logical name RDMS$SET_FLAGS or using the SET FLAGS statement with the string 'HASHING(JOINS)'. See Section 3.1.10 for more details.
- The new JOIN BY HASH clause in CREATE OUTLINE statement and the OPTIMIZE OUTLINE clause of the select statement. See Section 3.1.11 for more details.
- Specifying the OPTIMIZE FOR HASH JOIN on the select statement.

```
SQL> select e.employee_id, (sh.salary_end - sh.salary_start) month (3)
cont> from employees e, salary_history sh
cont> where e.employee_id = sh.employee_id
cont>       and e.employee_id <= '00164'
cont>       and sh.salary_end is not null
cont> optimize for hash join
cont> ;
.
.
.
```

Example

This example shows the strategy used by the optimizer when HASH JOIN is enabled.

```
SQL> set flags 'STRATEGY,DETAIL(2)';
SQL> set flags 'HASHING(JOINS)';
SQL>
SQL> select e.employee_id, (sh.salary_end - sh.salary_start) month (3)
cont> from employees e, salary_history sh
cont> where e.employee_id = sh.employee_id
cont>       and e.employee_id <= '00164'
cont>       and sh.salary_end is not null
cont> ;
Tables:
  0 = EMPLOYEES
  1 = SALARY_HISTORY
Conjunct: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
Hash Q1
  Outer Build
  Match_Key:0.EMPLOYEE_ID
    Index only retrieval of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:1]
        Keys: 0.EMPLOYEE_ID <= '00164'
  Inner Probe
  Match_Key:1.EMPLOYEE_ID
    Conjunct: NOT MISSING (1.SALARY_END)
    Conjunct: 1.EMPLOYEE_ID <= '00164'
    Get     Retrieval by index of relation 1:SALARY_HISTORY
      Index name  SH_EMPLOYEE_ID [0:1]
        Keys: 1.EMPLOYEE_ID <= '00164'
Table=0:EMPLOYEES #Buckets=47 #Hits=1 #Collisions=0 #Dups=0 #Dups_Chain=0
Load_Factor=   2.127659574468085E-002
 E.EMPLOYEE_ID
 00164            008
 00164            006
 00164            016
3 rows selected
SQL>
```

Now try the same query with NOHASHING.

```
SQL> set flags 'NOHASHING(JOINS)';
SQL>
SQL> select e.employee_id, (sh.salary_end - sh.salary_start) month (3)
cont> from employees e, salary_history sh
cont> where e.employee_id = sh.employee_id
cont>       and e.employee_id <= '00164'
cont>       and sh.salary_end is not null
cont> ;
Tables:
  0 = EMPLOYEES
  1 = SALARY_HISTORY
Conjunct: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
Match  Q1
  Outer loop      (zig-zag)
  Match_Key:0.EMPLOYEE_ID
  Index_Key:EMPLOYEE_ID
    Index only retrieval of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:1]
        Keys: 0.EMPLOYEE_ID <= '00164'
  Inner loop      (zig-zag)
  Match_Key:1.EMPLOYEE_ID
  Index_Key:EMPLOYEE_ID
    Conjunct: NOT MISSING (1.SALARY_END)
    Conjunct: 1.EMPLOYEE_ID <= '00164'
    Get     Retrieval by index of relation 1:SALARY_HISTORY
      Index name  SH_EMPLOYEE_ID [0:1]
        Keys: 1.EMPLOYEE_ID <= '00164'
 E.EMPLOYEE_ID
 00164            008
 00164            006
 00164            016
3 rows selected
SQL>
```

3.1.13 New ALTER DATABASE ... LOAD ACL IDENTIFIERS Clause

In this release of Oracle Rdb, the database administrator can automatically and simply create users and roles in the database. This new clause, LOAD ACL IDENTIFIERS, is part of the ALTER DATABASE statement and can be run as often as necessary to add new users and roles derived from the existing access control lists (ACLs) granted to the database and database objects.

The following example shows this clause on a sample database:

```
SQL> attach 'filename WAREHOUSE';
SQL>
SQL> show roles;
Roles in database with filename WAREHOUSE
 No roles found
SQL> show users;
Users in database with filename WAREHOUSE
 No users found
SQL>
SQL> disconnect all;
SQL>
SQL> alter database
cont>     filename WAREHOUSE
cont>
cont>     load acl identifiers
cont> ;
SQL>
SQL> attach 'filename WAREHOUSE';
SQL>
SQL> show roles;
Roles in database with filename WAREHOUSE
     CDD$EXTENDER
     CDD$SYSTEM
     STORES_CUST_READABLE
     STORES_EXTRACT_TEXT
     STORES_MAIL_TEXT
     STORES_PRINT_TEXT
     STORES_USER
SQL> show users;
Users in database with filename WAREHOUSE
     FLEE
     ISMITH
     JJONES
     KSTJOHN
     WH_QUERY_1
     WH_QUERY_2
     WH_QUERY_3
     WH_QUERY_4
     WH_QUERY_5
SQL>
SQL> disconnect all;
SQL>
```

When the database administrator uses the GRANT statement to give access to users and OpenVMS rights identifiers (aka roles), they are recorded in the access control lists for each object; database, table, view, column, sequence, module, and routine. This clause of the ALTER DATABASE statement reads every ACL in the database and creates USER and ROLE definitions if necessary.

Note

Some access control entries (ACEs) may use OpenVMS group identifiers (PUBLIC, [*,*], [ADMIN,*], [*], etc), or special modifier rights identifiers (BATCH, DIALUP, INTERACTIVE, LOCAL, NETWORK, REMOTE) which are not valid users and roles - these will be ignored by the LOAD ACL IDENTIFIERS clause.

In addition to the DBADM privilege required to use ALTER DATABASE, this clause also requires SECURITY on the database. Alternately, the user must be granted the ALTER ANY DATABASE, CREATE ANY USER and CREATE ANY ROLE database system privilege.

3.1.14 New ALTER TABLE Actions for READ ONLY Table

This release of Oracle Rdb adds the ability to change a table to READ ONLY access. Once committed, no other application or interactive SQL session may modify rows in the table. You can issue database definition statements (DDL) as long as they do not modify any table data. Operations on indices associated with the table are allowed when the table is in READ ONLY mode. To revert to a read-write table, the clause READ WRITE can be applied.

While the table is READ ONLY the following restrictions apply:

- The data manipulation statements INSERT, UPDATE, DELETE may not modify rows in the table.
- Table updates via LIST cursor may fail during the OPEN or CLOSE statement depending on the cursor declaration.
- The SELECT statement using the FOR UPDATE clause may fail because it tries to apply UPDATE semantics.

```
SQL> select * from SAMPLE_TABLE for update;
%RDB-E-READ_ONLY_REL, relation SAMPLE_TABLE was reserved for read access;
updates not allowed
```

- The TRUNCATE TABLE statement is not permitted to truncate rows from the table.
- ALTER TABLE ... ADD COLUMN is permitted unless a DEFAULT is added for the new column (either explicitly or implicitly from a domain reference). In this case, Rdb would normally execute an UPDATE statement to include the default into each pre-existing row.
- Most other database definition statements are permitted. For instance, CREATE INDEX, ALTER INDEX ... REBUILD ALL PARTITIONS, DROP INDEX, can all be performed while the table is in this state.

The following example shows the diagnostic reported by Oracle Rdb.

```
SQL> alter table SAMPLE_TABLE
cont>    read only
cont> ;
SQL>
SQL> show table (comment) SAMPLE_TABLE;
Information for table SAMPLE_TABLE

Comment on table SAMPLE_TABLE:
Samples table

Table is set READ ONLY

SQL> truncate table SAMPLE_TABLE;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-READ_ONLY_REL, relation SAMPLE_TABLE was reserved for read access;
updates not allowed
SQL> commit;
SQL>
```

3.1.15 New NULLS FIRST and NULLS LAST Options for ORDER BY Clause

This release of Oracle Rdb adds the NULLS FIRST and NULLS LAST options to the ORDER BY clause. These options control the ordering of the NULL values relative to the ordering of the key data. This is demonstrated by the four examples shown below.

```
SQL> select employee_id, salary_start, salary_end, salary_amount
cont>  from salary_history sh
cont>  where employee_id = '00164'
cont>  order by salary_end asc nulls first
cont> ;
 EMPLOYEE_ID   SALARY_START   SALARY_END    SALARY_AMOUNT
 00164         14-Jan-1983    NULL             $51,712.00
 00164          5-Jul-1980     2-Mar-1981      $26,291.00
 00164          2-Mar-1981    21-Sep-1981      $26,291.00
 00164         21-Sep-1981    14-Jan-1983      $50,000.00
4 rows selected
SQL>
SQL> select employee_id, salary_start, salary_end, salary_amount
cont>  from salary_history sh
cont>  where employee_id = '00164'
cont>  order by salary_end asc nulls last
cont> ;
 EMPLOYEE_ID   SALARY_START   SALARY_END    SALARY_AMOUNT
 00164          5-Jul-1980     2-Mar-1981      $26,291.00
 00164          2-Mar-1981    21-Sep-1981      $26,291.00
 00164         21-Sep-1981    14-Jan-1983      $50,000.00
 00164         14-Jan-1983    NULL             $51,712.00
4 rows selected
SQL>
SQL> select employee_id, salary_start, salary_end, salary_amount
cont>  from salary_history sh
cont>  where employee_id = '00164'
cont>  order by salary_end desc nulls first
cont> ;
 EMPLOYEE_ID   SALARY_START   SALARY_END    SALARY_AMOUNT
 00164         14-Jan-1983    NULL             $51,712.00
 00164         21-Sep-1981    14-Jan-1983      $50,000.00
 00164          2-Mar-1981    21-Sep-1981      $26,291.00
 00164          5-Jul-1980     2-Mar-1981      $26,291.00
4 rows selected
SQL>
SQL> select employee_id, salary_start, salary_end, salary_amount
cont>  from salary_history sh
cont>  where employee_id = '00164'
cont>  order by salary_end desc nulls last
cont> ;
 EMPLOYEE_ID   SALARY_START   SALARY_END    SALARY_AMOUNT
 00164         21-Sep-1981    14-Jan-1983      $50,000.00
 00164          2-Mar-1981    21-Sep-1981      $26,291.00
 00164          5-Jul-1980     2-Mar-1981      $26,291.00
 00164         14-Jan-1983    NULL             $51,712.00
4 rows selected
SQL>
```

3.1.16 Enhancements to RMU Unload After_Image (LogMiner) Interface

This release of Oracle Rdb adds support for the following new options for the RMU Unload After_Image (aka LogMiner) command.

- The RMU Unload After_Image command can now output an XML script containing the updates to the selected table or tables.
- When the FORMAT qualifier specifies one of DELIMITED_TEXT, DUMP or XML, the TRIM option can also be used to trim leading and/or trailing spaces.
- The SYMBOLS qualifier now accepts the keyword LOCAL (default) or GLOBAL. The default behavior is to generate local scope DCL symbols. When Symbols=GLOBAL is used, these symbols have global scope.

3.1.16.1 New XML Option to FORMAT Qualifier

When using FORMAT=XML, the following options can also be specified:

- CHARACTER_ENCODING_XML

  When using RMU Unload After_Image Format=XML, the XML header record will, by default, use the character encoding "ISO-8859-1". For example, this will appear in the header of the XML file.

  ```
  <?xml version="1.0" encoding="ISO-8859-1"?>
  ```

  This encoding (ISO-8859-1) is Latin 1 and covers encoding of many European character sets. However, this encoding is not adequate if you use other character encoding for Asian languages, or languages not covered by this ISO Standard.

  This release of Oracle Rdb adds an option, CHARACTER_ENCODING_ XML, that allows the command procedure to specify an alternate character encoding. For example, if you wish to have the XML header describe UTF8, then specify the qualifier /FORMAT=(XML,CHAR="utf-8").

  ```
  <?xml version="1.0" encoding="utf-8"?>
  ```

- DATA_XML_NULL

  This option accepts one of the following keywords which control the output of NULL column values: DROP, NIL_ATTRIBUTE, or EMPTY. If this option is not specified, the default is EMPTY as shown in the following example.

  This is a fragment of the XML data generated by RMU Unload After_Image with the qualifier /FORMAT=(XML,TRIM=TRAILING) defaulting to DATA_ XML_NULL=EMPTY.

  ```
  <ROW>
   <RDB_LM_ACTION>M</RDB_LM_ACTION>
   <RDB_LM_RELATION_NAME>SAMPLE2</RDB_LM_RELATION_NAME>
   <RDB_LM_RECORD_TYPE>32</RDB_LM_RECORD_TYPE>
   <RDB_LM_DATA_LEN>65254</RDB_LM_DATA_LEN>
   <RDB_LM_NBV_LEN>3</RDB_LM_NBV_LEN>
   <RDB_LM_DBK>60:14417:0</RDB_LM_DBK>
   <RDB_LM_START_TAD>2020-05-11T10:20:50.88</RDB_LM_START_TAD>
   <RDB_LM_COMMIT_TAD>2020-05-11T10:20:50.89</RDB_LM_COMMIT_TAD>
   <RDB_LM_TSN>712</RDB_LM_TSN>
   <RDB_LM_RECORD_VERSION>1</RDB_LM_RECORD_VERSION>
   <IDENT>101</IDENT>
   <COMMENT/>
   <DETAILS/>
  </ROW>
  ```

If DATA_XML_NULL is specified as DROP, then that column's value is omitted from the XML record. If DATA_XML_NULL is specified as NIL_ ATTRIBUTE, then the XML tag attribute for the column is output as xsi:nil="true".

- TRIM

  This option allows the data values for the columns to have trailing and leading spaces and horizontal tab characters removed from the columns.

3.1.16.2 TRIM Option

When the FORMAT selected is one of DELIMITED_TEXT, DUMP or XML, then RMU can be instructed to trim trailing and/or leading spaces and horizontal tab characters from the columns.

- The default when FORMAT is XML or DELIMITED_TEXT is no trimming. The default when FORMAT is DUMP is TRIM_TRAILING.
- The TRIM option is not compatible with FORMAT=BINARY and FORMAT=TEXT.
- One of the following keywords can be specified for TRIM: TRAILING, LEADING and BOTH. If TRIM is specified without qualification, then TRAILING is assumed.

The following example shows the use of format XML with the TRIM=BOTH option.

```
$       RMU/UNLOAD-
            /AFTER_IMAGE -
            /INCLUDE=ACTION:(COMMIT,DELETE,NOMODIFY)-
            /LOG-
            /TABLE=(name=SAMPLE2, output=SAMPLE3.DAT)-
            /FORMAT=(XML,TRIM=BOTH)-
            /ORDER_AIJ_FILES-
        USER1:[TESTER.LOGMINER]LOGMINER_DB -
        USER1:[TESTER.LOGMINER]AIJ_BU_*.BAIJ
$
```

Usage Notes

- When unloading rows with columns that have many trailing spaces, then using FORMAT=(XML,TRIM) or FORMAT=(DELIMITED_TEXT,TRIM) can, in some cases, reduce the size of the output file without loss of significant data.
- When dumping rows with long columns that have many trailing spaces, then using /FORMAT=(DUMP,TRIM) can significantly reduce the size of the output file. Therefore, RMU Unload After_Image implicitly enables TRIM=TRAILING.

3.1.16.3 SYMBOLS Qualifier

The SYMBOLS qualifier now accepts the keyword LOCAL (default) or GLOBAL. The default behavior is to generate local scope DCL symbols. When Symbols=GLOBAL is used, these symbols have global scope.

Arguments

Symbols
Symbols=LOCAL
Symbols=GLOBAL
NoSymbols

Specifies whether DCL symbols are to be created, indicating information about records extracted for each table.

The default is Symbols, which causes local symbols to be created. Use Symbols=GLOBAL to have RMU define global symbols. Use NoSymbols to prevent creation of any DCL symbols.

If a large number of tables are being unloaded, too many associated symbols may be created and the CLI symbol table space can become exhausted. The error message "LIB-F-INSCLIMEM, insufficient CLI memory" is returned in this case. Specify the Nosymbols qualifier to prevent creation of the symbols.

3.1.17 New Named Partition Support for RESERVING Clause

This release of Oracle Rdb adds the ability to use named partitions in the RESERVING clause of the SET TRANSACTION or DECLARE TRANSACTION statements. In prior versions, only partition numbers were allowed.

The partition names might be system generated (as shown below for the EMPLOYEES_MAP from the MF_PERSONNEL database) or they can be defined as part of the CREATE STORAGE MAP statement.

The following example shows the partition numbers as well as the system generated partition names under the *Partition information for storage map* output.

```
SQL> show storage map employees_map
     EMPLOYEES_MAP
 For Table:         EMPLOYEES
 Placement Via Index:    EMPLOYEES_HASH
 Partitioning is:    UPDATABLE

 Partition information for storage map:
 Compression is:    ENABLED
  Partition: (1) SYS_P00079
   Storage Area: EMPIDS_LOW
  Partition: (2) SYS_P00080
   Storage Area: EMPIDS_MID
  Partition: (3) SYS_P00081
   Storage Area: EMPIDS_OVER

SQL>
```

Usage Notes

- The PARTITION clause accepts a list of partition names or a list of partition ordinal values. You may not mix numeric and named notations.
- Duplicate partition names in the RESERVING clause will cause an exception. Review the RESERVING clause and correct the partition names.

```
set transaction
    read write
    evaluating
        job_history_foreign1 at verb time
        ,salary_history_foreign1 at verb time
    reserving
        employees partition (SYS_P00080, SYS_P00080) for exclusive write
;
%RDB-E-BAD_TPB_CONTENT, invalid transaction parameters in the transaction
parameter block (TPB)
-RDMS-E-DUPPARTNAME, partition SYS_P00080 for table EMPLOYEES already used
```

– Unknown partition names in the RESERVING clause (which might occur due to a change in the storage map definition) will cause an exception. Use the SHOW STORAGE MAP statement to review the partition names.

```
create module mod_testing1a
 language sql
 procedure proc_xa ();
 begin not atomic
 set transaction
    read write
    evaluating salary_history_foreign1 at verb time
    reserving employees partition (SYS_P00080, "UNKNOWN", SYS_P00081)
              ,departments for protected write;
  commit;
 end;
end module;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-BAD_TPB_CONTENT, invalid transaction parameters in the transaction
parameter block (TPB)
-RDMS-F-PARTNEXTS, partition "UNKNOWN" does not exist in this map or index
"EMPLOYEES_MAP"
```

3.1.18 RMU Backup No Longer Supports HUFFMAN or LZSS Compression, Use ZLIB Instead

This release of Oracle Rdb removes the compression options HUFFMAN and LZSS from the RMU Backup and RMU Backup After_Journal commands.

These older compression algorithms are much slower than the default ZLIB compression. Orders of magnitude more CPU is required in some cases. If you receive a diagnostic as shown in the following example, then Oracle recommends accepting the default, changing the DCL command procedure, or RMU PLAN file to explicitly state ZLIB. Additionally, ZLIB compression allows the database administrator to determine levels of compression efficiency, from less time to more effective compression. Refer to Oracle Rdb RMU Reference Manual for more details.

The following example shows the new diagnostic reported by RMU.

```
$ rmu/backup-
    /disk_file-
    /list_plan=plan_1.plan-
    /execute-
    /compress=lzss-
    /parallel=Executor_Count=3 -
    sql$database -
    sav_l1.rbf,sav_l2.rbf,sav_l3.rbf
%RMU-E-NOTSUPFORVER, The function COMPRESSION.LZSS is not supported for Oracle
Rdb V7.4-100
-RMU-I-COMPUSEZLIB, use the default, or specify ZLIB compression
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 18-JUN-2020 15:45:23.27
$
```

Also note that neither HUFFMAN nor LZSS are accepted by the RMU Set After_ Journal Backups qualifier. That command has always required the preferred ZLIB algorithm for compression.

4

Documentation Corrections, Additions and Changes

This chapter provides corrections for documentation errors and omissions.

4.1 Documentation Corrections

4.1.1 Oracle Rdb Release 7.4.x.x New Features Document Added

A new document has been created which contains all of the New Features Chapters from all Rdb 7.4 Release Notes. This document will be included in saveset A of the Rdb kit. It is called RDB_NEWFEATURES_74xx and will be available in postscript, text and PDF format. This will provide customers with one document to reference to find out about all new features that have been added to the Rdb 7.4 releases.

4.1.2 New Optional Builtin Function RDB$$IS_ROW_FRAGMENTED

Bug 3788472

This release note was in the Rdb 7.1.3 Release Notes but was inadvertently left out of the SQL Reference Manual.

This release of Oracle Rdb supports a new optional builtin function that can determine if a row is fragmented. The function, RDB$$IS_ROW_FRAGMENTED must be declared as function using the attributes and properties as shown below.

```
declare function RDB$$IS_ROW_FRAGMENTED
    (in :dbk char(8) character set unspecified)
    returns integer;
```

The following example shows the usage on the WORK_STATUS table in the PERSONNEL database.

```
SQL> declare function RDB$$IS_ROW_FRAGMENTED
cont>    (in :dbk char(8) character set unspecified)
cont>    returns integer;
SQL>
SQL> select dbkey, RDB$$IS_ROW_FRAGMENTED (dbkey) from work_status;
                  DBKEY
              99:10:12            0
              99:10:13            0
              99:10:14            0
3 rows selected
```

Usage Notes

- This routine may only be used from Interactive and Dynamic SQL.
- Only valid DBKEY values should be passed to the function.
- If the DBKEY passed is not the current row, then additional I/O may be required to fetch the target row.

- If the DBKEY is for a vertically partitioned table, then only the fragmented state of the primary segment is reported. There is currently no programatic method to determine fragmented secondary segments.
- Temporary table and information table rows are never fragmented as they reside in virtual memory only.
- Fragmentation occurs when either the row is too large to fit entirely on a page or an existing row was updated to a larger size and no space existed at that time for the expanded row. The first case requires that the page size be changed for the area. However, for the second case, a DELETE and INSERT of the row might remove the fragmentation. In that case, this function allows the DBA to identify candidate fragmented rows. Fragmentation may occur when compression is enabled and the compressed row size changes due to changed data, NULL values replaced with non-NULL values, or ALTER TABLE or ALTER DOMAIN statements that have increased the size of columns.

4.1.3 Undocumented /TRANSACTION=EXCLUSIVE Option for RMU Populate_Cache

The EXCLUSIVE option was not documented in prior releases of Oracle Rdb. This is a revised section for the RMU Populate_Cache Command and will appear in a future update to the Oracle RMU Reference Manual.

- Transaction_Type=option

  Allows you to specify the transaction mode for the transactions used to perform the populate cache operation. Valid options are:

  - Automatic
  - Exclusive
  - Read_Only
  - Noread_Only

  You must specify an option if you use this qualifier.

  If you do not specify this qualifier, the default is Transaction_Type=Automatic. This qualifier specifies that Oracle RMU is to determine the transaction mode. For example, if READ ONLY transactions can not be performed, an equivalent READ WRITE transaction will be started.

  The Transaction_Type=Read_Only qualifier specifies the transactions used to perform the analyze operation be set to read-only mode.

  The Transaction_Type=Noread_Only qualifier specifies that the transactions used for the analyze operation be set to read/write mode. You might select this option if you want to avoid the growth of snapshot files that occurs during a long running read-only transaction and are willing to incur the cost of increased locking that occurs during a read/write transaction.

  The Transaction_Type=Exclusive readies all accessed storage areas in EXCLUSIVE mode. Therefore, it avoids the growth of snapshot files (no updates will be made to any snapshot file) and will use limited locking.

  Note

  Using Transaction_Type=Exclusive might interfere with other actions occurring on the database; especially note that ONLINE backup and some

actions of the Row Cache server are incompatible with this Transaction_ Type.

4.1.4 Action of DECLARE TRANSACTION When SQL$DATABASE is Defined

Bug 5705204

The following clarification is missing from the DECLARE TRANSACTION Statement documentation in the Oracle Rdb SQL Reference Manual.

- In Dynamic SQL and Interactive SQL, the DECLARE TRANSACTION statement is applied to all attached databases. When those databases are disconnected, the declared default transaction is discarded. If there are no attached databases, then the DECLARE TRANSACTION is used as a session wide default.

  When the first executable statement in a Dynamic SQL or an Interactive SQL session is a DECLARE TRANSACTION statement and the logical name SQL$DATABASE is defined, then the DECLARE statement will implicitly attach to that database. The declared default transaction is applied to that database attach. A subsequent DISCONNECT statement will disconnect from the database and discard the default transaction.

4.1.5 RDB$USAGE Field Values

The following information is missing from the Rdb SQL Reference Manual.

The table Rdb$INTERRELATIONS records much of the dependency information when one object references another in the database. Such information is used by DROP ... RESTRICT statements to prevent an object being deleted when it is required by some other object. For instance, a function may use one or more columns from a table in a query. That table and its columns will be recorded with a value in RDB$USAGE of 'Storage Map'.

Many reported errors include text from the RDB$USAGE field to explain the type of dependency preventing the DROP from succeeding. These text strings are described in Table 4–1, Rdb$USAGE Field Values.

Table 4–1 Rdb$USAGE Field Values

Field Value	Description
Computed Field	A Computed by or Automatic column references this table, view, column or function
Constraint	Constraint definition references table, view, column, sequence or function
Storage Map	Storage map references table and column
View	View definition requires table, view, column, sequence or function
View Field	View column requires table, view, column, sequence or function
Trigger	Trigger definition requires table, view, column, sequence or function

(continued on next page)

Table 4–1 (Cont.) Rdb$USAGE Field Values

Field Value	Description
RelConstraint	A table (relation) constraint references a table
Domain Constraint (VALID IF)	A domain constraint (or VALID IF) references this routine or sequence
Requires	This table, temporary table (with module name), or index is used by a query outline
Procedure	Procedure definition requires table, view, column, sequence or function
Function	Function definition requires table, view, column, sequence or function
Default Txn Reserving	A stored module DECLARE TRANSACTION references a table or view in the RESERVING clause
Default Txn Evaluating	A stored module DECLARE TRANSACTION references a constraint in the EVALUATING clause
Lang Semantics	A stored function, procedure or trigger uses wildcard for column list. This includes SELECT *, or INSERT with an omitted column list
Cast As Domain	A CAST function referenced a domain name
Temp Table Using Domain	A DECLARE LOCAL TEMPORARY TABLE used a domain for a columns data type
Computed Column in Temp Table	A computed by or automatic column defined by a DECLARE LOCAL TEMPORARY TABLE or DECLARE LOCAL TEMPORARY VIEW references this object
Module Variable Default Value	A module global DECLARE statement used a DEFAULT clause. Table, view, function, domain and sequence dependencies are recorded
Referenced by Synonym	When a synonym is created, a dependency is stored
Default Value	A table column uses a DEFAULT clause. Table, view, function, domain and sequence dependencies are recorded
Constraint Index	When SET FLAGS 'AUTO_INDEX' is active, any constraint definition will define an index matching the columns of the constraint
Module Variable	This module variable uses this domain
Routine Parameter	Not currently used. Reserved for future use
Temp Table Reference	A DECLARE LOCAL TEMPORARY TABLE references this table in the LIKE clause
Storage Map Function	When CREATE STORAGE MAP is executed, a system routine is created to reflect the mapping. Those column dependencies are recorded

4.1.6 Updated Documentation for RDMS$BIND_CODE_OPTIMIZATION Logical

The logical name RDMS$BIND_CODE_OPTIMIZATION has different uses on Alpha and Integrity systems.

- Alpha Systems

In prior releases of Oracle Rdb, queries and procedures were rendered as small Alpha instruction sequences which were used at runtime to implement all or part of the query strategy. This meant that executable code and data could reside on any Alpha page (8192 byte) sections.

Recent performance testing shows that OpenVMS for Alpha running on an Alpha emulator (such as Stromasys, Inc's CHARON-AXP) can greatly benefit from isolating the generated code instructions from updatable data areas. This change in memory management can be enabled by defining this logical to the value 3. Please note that such systems will typically require more virtual memory for the alternate memory management.

This logical name can be defined to the value (0) to disable the alternate memory management or defined as (3) to enable the alternate memory management. The default is 0 on Alpha systems and other values are undefined.

For more information on Stromays, Inc, please check their web site at: https://www.stromasys.com/solutions/charon-axp/

- Integrity systems

  When Oracle Rdb first became available on the Integrity platform, it used a portable instruction interpreter. Now, by default, Oracle Rdb generates native IA64 instructions to implement part of the query and procedure execution code. This native code execution in general delivers improved performance but at the expense of more virtual memory usage.

  This logical name can be defined to the value (0) to disable the native instruction compiler and use the interpreted instructions, or defined as (2) to enable the native IA64 instruction generator. The default is 2 on Integrity systems and other values are undefined.

The RMU Show Logical_Names command can be used to display the latest documentation for this and other logical names used by Oracle Rdb.

```
$ rmu /show Logical_Names RDMS$BIND_CODE_OPTIMIZATION/Description
  "RDMS$BIND_CODE_OPTIMIZATION" = Undefined
...etc...
```

4.1.7 New RDM$BIND_AIJ_BUFFER_POOL_COUNT Logical

The description of this new logical was inadvertently left out of the Rdb Release 7.3.1.3 Release Notes.

A new RDM$BIND_AIJ_BUFFER_POOL_COUNT logical has been added for use by various components of Oracle Rdb. This logical name can be used to change the number of buffers in the internal buffer pool used by the RMU/UNLOAD AFTER_JOURNAL command to select and process journal records from database transactions. If the logical name is not defined, a default of 10 buffers will be used.

The RDM$BIND_AIJ_BUFFER_POOL_COUNT logical allows a whole integer number of buffers between 4 and 16 to be specified for the RMU/UNLOAD AFTER_JOURNAL buffer pool. The current value of the RDM$BIND_AIJ_BUFFER_POOL_COUNT logical can be displayed using the RMU/SHOW LOGICAL_NAMES command if it is defined as a system logical.

The following examples show that the RMU/SHOW LOGICAL_NAMES command can be used to show if RDM$BIND_AIJ_BUFFER_POOL_COUNT is currently defined as a system logical. They also show that RDM$BIND_AIJ_BUFFER_POOL_COUNT can be defined as a process logical, and that if a value is defined for RDM$BIND_AIJ_BUFFER_POOL_COUNT that is not a whole integer between 4 and 16, an error will be returned by the RMU/UNLOAD AFTER_JOURNAL command and the RMU/UNLOAD AFTER_JOURNAL command will be aborted without processing any After Image Journal records.

```
$ RMU/SHOW LOGICAL_NAMES RDM$BIND_AIJ_BUFFER_POOL_COUNT
$
$!
$ RMU/SHOW LOGICAL_NAMES/UNDEFINED RDM$BIND_AIJ_BUFFER_POOL_COUNT
  "RDM$BIND_AIJ_BUFFER_POOL_COUNT" = Undefined
$!
$ DEFINE/SYSTEM RDM$BIND_AIJ_BUFFER_POOL_COUNT 4
$ RMU/SHOW LOGICAL_NAMES RDM$BIND_AIJ_BUFFER_POOL_COUNT
  "RDM$BIND_AIJ_BUFFER_POOL_COUNT" = "4" (LNM$SYSTEM_TABLE)
$!
$ DEFINE/SYSTEM RDM$BIND_AIJ_BUFFER_POOL_COUNT 16
%DCL-I-SUPERSEDE, previous value of RDM$BIND_AIJ_BUFFER_POOL_COUNT has been
superseded
$ RMU/SHOW LOGICAL_NAMES RDM$BIND_AIJ_BUFFER_POOL_COUNT
  "RDM$BIND_AIJ_BUFFER_POOL_COUNT" = "16" (LNM$SYSTEM_TABLE)
$!
$ DEFINE RDM$BIND_AIJ_BUFFER_POOL_COUNT 3
$ RMU/UNLOAD/AFTER_JOURNAL/INCL=ACT=(NOCOMMIT)/STAT=3600/LOG -
/RESTORE_METADATA=LOGMINER_METADATA.TXT -
/TABLE=(NAME=TEST_TABLE,OUTPUT=RDB_LOGMINER_OUTPUT_FILE) -
TEST.AIJ
%RMU-I-LMMFRDCNT, Read 13542 objects from metadata file
"DEVICE:[DIRECTORY]LOGMINER_METADATA.TXT;1"
%RMU-I-UNLAIJFL, Unloading table TEST_TABLE to
DEVICE:[DIRECTORY]RDB_LOGMINER_OUTPUT_FILE.DAT;1
%RMU-F-BADBNDPRM, bad bind parameter RDM$BIND_AIJ_BUFFER_POOL_COUNT value "3"
-RMU-F-BADBOUNDS, value not in range 4 to 16
%RMU-F-FTL_RMU, Fatal error for RMU operation at 17-MAR-2019 09:58:08.42
------------------------------------------------------------------------------
 ELAPSED:    0 00:00:00.00  CPU: 0:00:00.00  BUFIO: 5  DIRIO: 0  FAULTS: 46
Table "TEST_TABLE" : 0 records written (0 modify, 0 delete)
Total : 0 records written (0 modify, 0 delete)
$!
$ DEFINE RDM$BIND_AIJ_BUFFER_POOL_COUNT 17
%DCL-I-SUPERSEDE, previous value of RDM$BIND_AIJ_BUFFER_POOL_COUNT has been
superseded
$ RMU/UNLOAD/AFTER_JOURNAL/INCL=ACT=(NOCOMMIT)/STAT=3600/LOG -
/RESTORE_METADATA=LOGMINER_METADATA.TXT -
/TABLE=(NAME=TEST_TABLE,OUTPUT=RDB_LOGMINER_OUTPUT_FILE) -
TEST.AIJ
%RMU-I-LMMFRDCNT, Read 13542 objects from metadata file
"DEVICE:[DIRECTORY]LOGMINER_METADATA.TXT;1"
%RMU-I-UNLAIJFL, Unloading table TEST_TABLE to
DEVICE:[DIRECTORY]RDB_LOGMINER_OUTPUT_FILE.DAT;2
%RMU-F-BADBNDPRM, bad bind parameter RDM$BIND_AIJ_BUFFER_POOL_COUNT value "17"
-RMU-F-BADBOUNDS, value not in range 4 to 16
%RMU-F-FTL_RMU, Fatal error for RMU operation at 17-MAR-2019 09:59:39.02
------------------------------------------------------------------------------
 ELAPSED:    0 00:00:00.00  CPU: 0:00:00.01  BUFIO: 5  DIRIO: 0  FAULTS: 46
Table "TEST_TABLE" : 0 records written (0 modify, 0 delete)
Total : 0 records written (0 modify, 0 delete)
$!
```

4.1.8 RDMSTT Image Optionally Installed

If you plan on using the cluster capability of RMU/SHOW STATISTICS, Oracle recommends that you install the RDMSTT74.EXE image on OpenVMS with the appropriate privileges.

The RMONSTART74.COM command procedure provided by the Oracle Rdb installation allows the system manager to optionally install the RDMSTT74.EXE image at monitor startup time. To take advantage of this, you will need to edit the SYS$STARTUP:RMONSTART74.COM procedure and remove the comment characters from the following lines:

```
$! CALL INSTALL_IMAGE SYS$COMMON:[SYSEXE]RDMSTT74.EXE
      "/OPEN/HEAD/PROT/PRIV=(CMKRNL,SYSPRV,SHARE)"
```

The command procedure SYS$STARTUP:RMONSTOP74.COM will automatically remove this image if it is installed.

4.1.9 Some Optional System Tables Can Be Relocated Using a User Defined Storage Map

All system tables are mapped by default to the system storage area RDB$SYSTEM. If the database is created with the DEFAULT STORAGE AREA clause, then some of these tables will automatically be created in the secondary system area. Additionally, there exists a set of optional system tables (which may not exist in all databases) which may be manually mapped to other storage areas.

To change the mapping for one (or more) of these system tables, you must follow these steps. See Table 4–2 for the list of these optional system tables that allow mapping.

1. Attach to the database. If you are creating a storage map for RDB$CATALOG_SCHEMA or RDB$SYNONYMS then you must attach with the option MULTISCHEMA IS OFF. You should not execute any queries on the database as this may cause the system table to be locked and prevent the CREATE and ALTER STORAGE MAP statements from completing.
2. Create a storage map for the optional system table. Note that only those listed here are able to be re-mapped and you must use the specified storage map names.

Table 4–2 Optional System Tables and Their Storage Map Names

Table Name	Storage Map Name	Associated Feature
RDB$CATALOG_ SCHEMA	RDB$CATALOG_ SCHEMA_MAP	Multischema databases
RDB$CHANGES	RDB$CHANGES_MAP	Replication Option for Rdb
RDB$CHANGES_MAX_ TSER	RDB$CHANGES_MAX_ TSER_MAP	Replication Option for Rdb
RDB$SYNONYMS	RDB$SYNONYMS_MAP	Multischema databases
RDB$TRANSFERS	RDB$TRANSFERS_MAP	Replication Option for Rdb
RDB$TRANSFER_ RELATIONS	RDB$TRANSFER_ RELATIONS_MAP	Replication Option for Rdb

(continued on next page)

Table 4–2 (Cont.) Optional System Tables and Their Storage Map Names

Table Name	Storage Map Name	Associated Feature
RDB$WORKLOAD	RDB$WORKLOAD_MAP	Workload Collection was Enabled

3. The storage map must be a simple storage map which simply describes the current state for this table, namely the name of the storage area in which the table resides. See the following example.

```
SQL> create storage map RDB$CHANGES_MAP
cont>   for RDB$CHANGES
cont>   store in RDB$SYSTEM;
```

   The following restrictions apply to the created storage map for these special system tables:

   - The storage map may not change the defaulted compression attributes
   - The storage map may not specify the logical area thresholds
   - The storage map may not be placed via an index
   - The storage map may not vertically partition the table
   - The storage map may only contain one storage area
   - And it must be mapped to the default storage area (this may be RDB$SYSTEM by default or the name of the user specified storage area using the DEFAULT STORAGE AREA clause during CREATE DATABASE)

4. Now that the storage map exists, you may use the ALTER STORAGE MAP statement to move the table to another area.

```
SQL> alter storage map RDB$CHANGES_MAP
cont>   store in RDB_CHANGES_AREA;
```

   The following restrictions apply to the altered storage map for these special system tables:

   - The storage map may not be placed via an index
   - The storage map may only contain one storage area
   - The storage map may not vertically partition the table
   - The ALTER STORAGE MAP operation may require exclusive access to the database as well as the table while the table data is relocated.

5. These storage map attributes for system tables are not currently exported by the SQL EXPORT DATABASE statement. Therefore, if you EXPORT and IMPORT your database, you will need to repeat these steps to re-map any of these system tables. It is expected that this restriction will be removed in a future version of Oracle Rdb.

4.1.10 Recovering an Oracle Rdb Database After RMU/RESTORE/ONLY_ROOT

Bug 12595718

If it is necessary to recover a database following the RMU/RESTORE/ONLY_ROOT command, be sure that the transaction state of the database is correctly set in the database root by the RMU/RESTORE/ONLY_ROOT command. Otherwise transactions in journal files created before the RMU/RESTORE/ONLY_ROOT command will be ignored by the RMU/RECOVER command, which uses the transaction state of the database stored in the database root file to select which journaled transactions to recover from the After Image Journal (AIJ) files used by the RMU/RECOVER command. Important journaled updates to database parameters, such as the client sequence numbers maintained both in the database root (CLTSEQ) and the database system tables (RDB$SEQUENCES), may be lost or made inconsistent.

The database should be verified both after the RMU/RESTORE/ONLY_ROOT command completes and after the RMU/RECOVER command completes. Please consult the documentation in the Oracle Rdb RMU Reference Manual for more information and examples of the options related to AIJ files and setting the database root transaction TSN and CSN values when the Rdb database root file (.rdb) is restored using the RMU/RESTORE/ONLY_ROOT command.

The TSN and CSN values set in the database root restored by the RMU/RESTORE/ONLY_ROOT command are displayed by an informational message if logging is enabled.

```
%RMU-I-SETRTTSNCSN, Setting Root Transaction Sequence TSN to 384,
 Commit Sequence CSN to 384
```

In this example, the /NOSET_TSN qualifier is used so that the TSN and CSN values of the restored root file are set to the values in the backup file used by the RMU/RESTORE/ONLY_ROOT command. As a result, the original journaled client sequence value of "21" is recovered by the RMU/RECOVER command executed after the RMU/RESTORE/ONLY_ROOT command.

```
$ sql$
SQL> attach 'file mf_personnel';
SQL> select RDB$SEQUENCE_NAME, RDB$NEXT_SEQUENCE_VALUE
cont> from rdb$sequences;
 RDB$SEQUENCE_NAME                RDB$NEXT_SEQUENCE_VALUE
 S                                                     21
1 row selected
exit;
$ delete mf_personnel.rdb;*
$ rmu/restore/only_root/log/NOSET_TSN mf_personnel
%RMU-I-AIJRSTBEG, restoring after-image journal "state" information
%RMU-I-AIJRSTJRN, restoring journal "AIJ1" information
%RMU-I-AIJRSTSEQ, journal sequence number is "0"
%RMU-I-AIJRSTSUC, journal "AIJ1" successfully restored from file
 "DEVICE:[DIRECTORY]AIJ_ONE.AIJ;1"
%RMU-I-AIJRSTJRN, restoring journal "AIJ2" information
%RMU-I-AIJRSTNMD, journal has not yet been modified
%RMU-I-AIJRSTSUC, journal "AIJ2" successfully restored from file
 "DEVICE:[DIRECTORY]AIJ_TWO.AIJ;1"
%RMU-I-AIJRSTEND, after-image journal "state" restoration complete
%RMU-I-SETRTTSNCSN, Setting Root Transaction Sequence TSN to 384,
 Commit Sequence CSN to 384
%RMU-I-AIJISON, after-image journaling has been enabled
%RMU-W-DOFULLBCK, full database backup should be done to ensure
 future recovery
%RMU-I-AIJRECEND, after-image journal "state" recovery complete
```

```
$
$ sql$
SQL> attach 'file mf_personnel';
SQL> select RDB$SEQUENCE_NAME, RDB$NEXT_SEQUENCE_VALUE
cont> from rdb$sequences;
 RDB$SEQUENCE_NAME                RDB$NEXT_SEQUENCE_VALUE
 S                                                      1
1 row selected
exit;
$ rmu/recover/out=recov.sav  AIJ_ONE.aij,AIJ_TWO.aij
$
$ sql$
SQL> attach 'file mf_personnel';
SQL> select RDB$SEQUENCE_NAME, RDB$NEXT_SEQUENCE_VALUE
cont> from rdb$sequences;
 RDB$SEQUENCE_NAME                RDB$NEXT_SEQUENCE_VALUE
 S                                                     21
1 row selected
exit;
$
```

4.1.11 Oracle Rdb Position on NFS Devices

This release note describes the supported usage of the NFS (Network File System) mounted devices by the Oracle Rdb product. NFS devices appear in most regards as local mounted file systems but do not allow the same level of sharing as provided by local OpenVMS devices. In addition, these files reside on a non-OpenVMS system (for instance a Linux or Windows system) and are therefore outside any scheme used by Rdb to lock buffers and pages of the database.

Active System Files

When Rdb is actively using database files, these files require specific sharing and locking to guarantee database integrity and recovery. Therefore, because of the limitations of the NFS mounted devices, active files such as the database root (.rdb), storage areas (.rda), snapshot files (.snp), row cache work file (.rdc), after image journal files (.aij), and before image recovery journal (.ruj) must not reside on an NFS mounted device.

Archived Data Files

Files that are not part of the active system may be stored on NFS mounted devices. For example, RMU /BACKUP /AFTER_JOURNAL can be used to archive an after image journal to a target on an NFS device. Similarly, RMU /BACKUP can perform a full or incremental backup to an Rdb backup file (.rbf) on an NFS device and RMU /RESTORE can use that NFS mounted source for database recovery, along with archived after image files from an NFS device processed by RMU /RECOVER.

Other Miscellaneous Files

Other files that might be used by an Rdb installation include options files, application procedures and sources, backup journals, record definitions files (.rrd), unloaded database files (.unl), exported databases (.rbr), log files, and so on. These sequential files may be stored on and referenced by RMU and SQL commands from an NFS mounted device.

Setting Up NFS

Complete instructions for setting up an NFS mounted device is beyond the scope of this release note and customers are directed to use system specific documentation for the server platform and for OpenVMS systems. However, during testing with Oracle Rdb we noted the need for the following qualifiers for the TCPIP MOUNT command.

- Use /ADF=CREATE. This ensures that attributes (such as block size and record length) are preserved on the server.
- Use /STRUCTURE=5. This will emulate an ODS-5 device and therefore allow the most complete OpenVMS Files-11 On-Disk Structure emulation.
- Use /TRANSPORT=UDP. For example,

```
$ tcpip mount dnfs1:/host="test.company.com"/path="/scratch"
        /stru=5/serve=unix/adf/vers=2/tran=udp
```

Read Performance Issues

In versions of Oracle Rdb prior to Rdb V7.3.1.2, a significant performance issue exists when reading sequential files from NFS mounted devices. Oracle Rdb uses the RMS read-ahead (RAH) attribute to improve sequential reads but this has an adverse effect when referencing an NFS device. The latest release of Oracle Rdb works around this issue by disabling the use of read-ahead when referencing an NFS device and would be the preferred version when using NFS devices.

Disclaimer

This information is provided to answer customer questions and should not be read as an endorsement or guarantee for NFS systems. Oracle expects configuration, functional testing, performance testing, security and integrity of the NFS data to be performed by our customers.

4.1.12 RDM$BIND_STAREA_EMERGENCY_DIR Logical Name

Bugs 19545970 and 3682207

RDM$BIND_STAREA_EMERGENCY_DIR is a HOT STANDBY logical name that can be utilized when replicating the creation of a new storage area from a master database to its standby database.

RDM$BIND_STAREA_EMERGENCY_DIR provides an alternate device and/or directory specification for the standby that can replace all or part of the master's file specification. Without the logical, the device and directory of the new storage area issued from the master must exist and match exactly on the standby. For example, on the master database we want to create a new starea, 1DGA11:[RDB_RANDOM.FOO]A1.RDA. We would issue the following command:

```
SQL>  alter database file rdb_random$db
        add storage area a1 filename $1$DGA11:[RDB_RANDOM.FOO]A1.RDA;
```

If the standby did not have a device called 1DGA11, the replication would fail and the AIJ Log Roll-Forward Server (LRS) logfile would log the failure.

```
3-SEP-2014 16:22:26.94 - Replicating master FILID 19
3-SEP-2014 16:22:26.94 - Attempting to create starea
"$1$DGA11:[RDB_RANDOM.FOO]A1.RDA;1" ALQ=2808
3-SEP-2014 16:22:26.95 - Unable to create storage area. STATUS: 00DDA89C
3-SEP-2014 16:22:26.95 - No emergency directory defined
3-SEP-2014 16:22:26.95 - Failure reason: LRSSRV$CREATE_AREA_CALLBACK - Could
not create storage area
```

Suppose the target disk on the standby was 1DGA109 and we defined the logical RDM$BIND_STAREA_EMERGENCY_DIR to point to that, as in the following example.

```
$ define/sys RDM$BIND_STAREA_EMERGENCY_DIR "$1$DGA109:"
$ create/dir $1$DGA109:[RDB_RANDOM.FOO]
```

The replication operation would succeed and the LRS logfile would show:

```
3-SEP-2014 15:42:45.65 - Attempting to create starea
"$1$DGA11:[RDB_RANDOM.FOO]A1.RDA;1" ALQ=2808
3-SEP-2014 15:42:45.67 - Unable to create storage area. STATUS: 00DDA89C
3-SEP-2014 15:42:45.67 - Using emergency area "$1$DGA109:[RDB_RANDOM.FOO]A1.RDA"
3-SEP-2014 15:42:45.67 - Attempting to create starea
"$1$DGA109:[RDB_RANDOM.FOO]A1.RDA" ALQ=2808
3-SEP-2014 15:42:45.68 - Starea creation successful
3-SEP-2014 15:42:45.70 - Attempting to create starea
"$1$DGA11:[RDB_RANDOM.FOO]A1.SNP;1" ALQ=404
3-SEP-2014 15:42:45.70 - Unable to create storage area. STATUS: 00DDA89C
3-SEP-2014 15:42:45.70 - Using emergency area "$1$DGA109:[RDB_RANDOM.FOO]A1.SNP"
3-SEP-2014 15:42:45.70 - Attempting to create starea
"$1$DGA109:[RDB_RANDOM.FOO]A1.SNP" ALQ=404
3-SEP-2014 15:42:45.71 - Starea creation successful
```

The RDM$BIND_STAREA_EMERGENCY_DIR logical must:

- Exist on the standby system prior to the create storage area operation.
- Be defined in the LNM$SYSTEM_TABLE table.
- Be a valid file specfication.

All standby databases on the node where the logical is defined share its use.

This logical was added back in Oracle Rdb Release 7.0.2 but the documentation of the logical was omitted.

4.1.13 RDMS-F-FULLAIJBKUP, Partially-Journaled Changes Made

Bug 7669735

The Oracle Rdb and Oracle CODASYL DBMS Guide to Hot Standby Databases states: "You can stop replication operations by explicitly entering the Replicate After_Journal Stop command on either the standby or master database nodes. Stopping replication on either database terminates replication on both databases."

Although the RMU/REPLICATE AFTER_JOURNAL STOP command may be issued against either Master or Standby to shut down replication, we have determined that there is at least one scenario where the choice is important relating to restarting replication in the future.

If you do the following, the operation will fail with a 'FULLAIJBKUP' error when starting the Master.

1. Stop replication on the Standby.
2. Set the old standby to be the new Master.
3. Set the old Master to be the new Standby.
4. Attempt to restart replication.

This is expected behavior. If the Standby is stopped prior to the Master, Oracle Rdb cannot determine if there has been any network traffic from the Master between the time that the Standby and Master shut down. Since any such information would be lost and may lead to data inconsistencies, replication will not be started.

The workaround for this scenario would be to stop replication on the Master, not the Standby. Consider the following two examples (assuming that replication is currently active):

Example 1: Initially stopping Replication on the Standby.

```
$! Stopping Replication on the Standby:

$ RMU/REPLICATE AFTER STOP/WAIT/LOG STANDBY$DB:STANDBY_PERSONNEL
%RMU-I-HOTSTOPWAIT, stopping database replication, please wait
%RMU-I-LOGMODSTR, stopped master database AIJ Log Replication Server

$! Start Replication of the Standby db (which was previously the Master)

$ RMU/REPLICATE AFTER_JOURNAL START MASTER:MF_PERSONNEL.RDB  -
     /CHECKPOINT=10 -
     /LOG           -
     /WAIT          -
     /BUFFERS=30    -
     /GAP_TIMEOUT=5 -
     /GOVERNOR=DISABLED -
     /MASTER_ROOT=STANDBY$DB:STANDBY_PERSONNEL.RDB -
     /ONLINE
%RMU-I-LOGMODSTR, started standby database AIJ Log Replication Server

$! Start Replication on the Master db (which was previously the Standby)

$ RMU/REPLICATE AFTER_JOURNAL START STANDBY$DB:STANDBY_PERSONNEL.RDB -
     /CHECKPOINT=100 -
     /LOG            -
     /WAIT           -
     /CONNECT_TIMEOUT=5 -
     /STANDBY_ROOT=MASTER:MF_PERSONNEL.RDB -
     /SYNCHRONIZATION=COLD  -
     /QUIET_POINT           -
     /TRANSPORT=TCPIP
%RMU-I-LOGMODSTR, started AIJ Log Server
%RDMS-F-CANTSTARTLCS, error starting AIJ Log Catch-Up Server process
-RDMS-F-FULLAIJBKUP, partially-journaled changes made; database may not be
recoverable
%RMU-F-FATALRDB, Fatal error while accessing Oracle Rdb.
%RMU-F-FTL_RMU, Fatal error for RMU operation at  4-AUG-2014 14:19:17.78
```

Example 2: Initially stopping Replication on the Master.

```
$! Stopping Replication on the Master:

$ RMU/REPLICATE AFTER STOP/WAIT/LOG MASTER$DB:MF_PERSONNEL.RDB
%RMU-I-HOTSTOPWAIT, stopping database replication, please wait
%RMU-I-LOGMODSTR, stopped master database AIJ Log Replication Server

$! Start Replication of the Standby db (which was previously the Master)
```

```
$ RMU/REPLICATE AFTER_JOURNAL START MASTER:MF_PERSONNEL.RDB  -
     /CHECKPOINT=10 -
     /LOG           -
     /WAIT          -
     /BUFFERS=30    -
     /GAP_TIMEOUT=5 -
     /GOVERNOR=DISABLED -
     /MASTER_ROOT=STANDBY$DB:STANDBY_PERSONNEL.RDB -
     /ONLINE
%RMU-I-LOGMODSTR, started standby database AIJ Log Replication Server

$! Start Replication on the Master db (which was previously the Standby)

$ RMU/REPLICATE AFTER_JOURNAL START STANDBY$DB:STANDBY_PERSONNEL.RDB -
     /CHECKPOINT=100 -
     /LOG            -
     /WAIT           -
     /CONNECT_TIMEOUT=5 -
     /STANDBY_ROOT=MASTER:MF_PERSONNEL.RDB -
     /SYNCHRONIZATION=COLD  -
     /QUIET_POINT           -
     /TRANSPORT=TCPIP
%RMU-I-LOGMODSTR, started AIJ Log Server
%RMU-I-LOGMODSTR, started master database AIJ Log Replication Server
```

The SYS$HELP:RMU_MSG*.DOC has more information about the FULLAIJBKUP error:

```
FULLAIJBKUP,   partially-journaled changes made; database may not be
               recoverable

Explanation:   Partially journalled changes have been made to the
               database.  This may result in the database being
               unrecoverable in the event of database failure; that
               is, it may be impossible to roll-forward the
               after-image journals, due to a transaction mis-match or
               attempts to modify objects that were not journaled.
               This condition typically occurs as a result of
               replicating database changes using the Hot Standby
               feature.

User Action:   IMMEDIATELY perform a full (not by-sequence)
               quiet-point AIJ backup to clear the AIJ journals,
               followed immediately by a full (no-quiet-point allowed)
               database backup.
```

4.1.14 Undocumented Hot Standby Logical Names

Bug 3264793

Table 4–3 Hot Standby Logical Names

Logical Name Description	Default Value	Minimum Value	Maximum Value
RDM$BIND_ALS_LOG_REOPEN_SECS Defines the number of seconds after which the ALS output file will automatically be reopened.	0 seconds (will not be reopened automatically)	0 seconds	31449600 (1 year)
RDM$BIND_ALS_LOG_REOPEN_SIZE Defines the number of blocks after which the ALS output file will automatically be reopened.	0 blocks (will not be reopened automatically)	0 blocks	Infinite

(continued on next page)

Table 4–3 (Cont.) Hot Standby Logical Names

Logical Name Description	Default Value	Minimum Value	Maximum Value
RDM$BIND_HOT_ABS_SUSPEND_ SHUTDOWN Defines whether or not the AIJ backup server (ABS) should be automatically suspended on graceful shutdown.	0	0	1
RDM$BIND_HOT_CHECKPOINT Specifies the number of messages per server checkpoint interval. If specified, the first threshold to be exceeded (message count or elapsed time) will cause the checkpoint.	100	1	50000
RDM$BIND_HOT_CHECKPOINT_INTERVAL Specifies a checkpoint interval, in minutes, to be used in addition to the /CHECKPOINT qualifier specified at Hot Standby startup. If specified, the first threshold to be exceeded (message count or elapsed time) will cause the LRS checkpoint.	0 minutes (don't use elapsed time)	0 minutes	10080 (7 days)
RDM$BIND_HOT_IGNORE_NET_TIMEOUT Specifies whether or not to ignore network timeout parameters if the LRS process is still active.	0	0	1
RDM$BIND_HOT_LOG_REOPEN_SECS Defines the number of seconds after which the AIJSERVER output file will automatically be reopened.	0 seconds (will not be reopened automatically)	0 seconds	604800 (1 week)
RDM$BIND_HOT_LOG_REOPEN_SIZE Defines the number of blocks after which the AIJSERVER output file will automatically be reopened.	0 blocks (will not be reopened automatically)	0	Infinite
RDM$BIND_HOT_NETWORK_ALT_NODE Defines the secondary network nodename to be used in the event of primary nodename network failure. This logical name allows you to specify an alternate routing pathway to the same standby database.	None		
RDM$BIND_HOT_NETWORK_RETRY Specifies a network retry timeout interval.	120 seconds	0	1800 (30 minutes)
RDM$BIND_LCS_AIJ_SCAN_IO_COUNT Defines the number of asynchronous I/O operations to be performed simultaneously during LCS catch-up.	64	1	128
RDM$BIND_LCS_LOG_REOPEN_SECS Defines the number of seconds after which the LCS output file will automatically be reopened.	0 seconds (will not be reopened automatically)	0 seconds	31449600 (1 year)
RDM$BIND_LCS_LOG_REOPEN_SIZE Defines the number of blocks after which the LCS output file will automatically be reopened.	0 blocks (will not be reopened automatically)	0 blocks	Infinite
RDM$BIND_LCS_QUIET_TIMEOUT Defines the number of seconds to wait for the LCS process to obtain the standby database quiet-point.	600 seconds	0 seconds (wait indefinitely)	Infinite

(continued on next page)

Table 4–3 (Cont.) Hot Standby Logical Names

Logical Name Description	Default Value	Minimum Value	Maximum Value
RDM$BIND_LCS_SYNC_COMMIT_MAX Defines the number of catch-up messages to synchronize with the standby database. A message may contain multiple transactions.	128 messages	32 messages	10000 messages
RDM$BIND_LRS_LOG_REOPEN_SECS Defines the number of seconds after which the LRS output file will automatically be reopened.	0 seconds (will not be reopened automatically)	0 seconds	31449600 (1 year)
RDM$BIND_LRS_LOG_REOPEN_SIZE Defines the number of blocks after which the LRS output file will automatically be reopened.	0 blocks (will not be reopened automatically)	0 blocks	Infinite
RDM$BIND_LRS_QUIET_TIMEOUT Defines the number of seconds to wait for the LRS process to obtain the standby database quiet-point.	600	0 seconds (wait indefinitely)	Infinite
RDM$BIND_STAREA_EMERGENCY_DIR Defines an alternate device and directory for the creation of storage areas on the standby database. The logical must be defined in the LNM$SYSTEM_TABLE table and it is shared by all standby databases on that node.			

4.1.15 Missing Documentation for the TRANSACTION_TYPE Keyword for GET DIAGNOSTICS

Prior versions of the SQL Reference Manual omitted the description of the TRANSACTION_TYPE keyword for GET DIAGNOSTICS.

TRANSACTION_TYPE returns the type of transaction being executed. The result will be one of the following strings: 'BATCH UPDATE', 'READ ONLY', 'READ WRITE', or 'NONE'.

The result data type is CHAR (31).

Example: Using TRANSACTION_TYPE to control actions of a procedure

Within a compound statement, you can use GET DIAGNOSTICS to retrieve information about the query state and its environment. In this example, we use the GET DIAGNOSTICS keywords TRANSACTION_TYPE and ROW_COUNT.

```
SQL> attach 'file MF_PERSONNEL';
SQL>
SQL> -- Sample procedure to use GET DIAGNOSTICS
SQL>
SQL> declare :rc integer;
SQL> declare :txn_type char(31);
SQL>
SQL> begin
cont>     set :rc = 0;
cont>     get diagnostics :txn_type = transaction_type;
cont>     trace '"' || :txn_type || '"';
cont>     case :txn_type
cont>         when 'BATCH UPDATE' then
cont>             begin
cont>             -- do nothing
cont>             end;
cont>         when 'READ ONLY' then
cont>             rollback;
cont>         when 'READ WRITE' then
cont>             delete from employees;
cont>             get diagnostics :rc = row_count;
cont>             trace 'Rows deleted = ', :rc;
cont>         when 'NONE' then
cont>             begin
cont>             -- no transaction so start one
cont>             set transaction read only;
cont>             end;
cont>     end case;
cont> end;
SQL>
SQL> print :txn_type, :rc;
 TXN_TYPE                                  RC
 NONE                                       0
SQL>
SQL> rollback;
```

4.1.16 Corrections to the EDIT STRING Documentation

Bugs 17365476 and 17365597

- The SQL Reference Manual, Volume 1, incorrectly stated that fields following a quoted literal would have leading zeros trimmed if the literal ended with a space. This was incorrect. The trimming only takes place after a space formatting character (B).

  This is the corrected text:

  Oracle Rdb automatically trims leading zeros from the first numeric field in the output, and any numeric field following a space formatting character (B). The year (Y) and fractional seconds (*) format fields are never trimmed of leading zeros.

- To have SQL represent an OpenVMS date format without removing the leading zero from the Hour field, use the literal string for space rather than the space formatting character (B).

  ```
  edit string 'YYYY-NN-DD" "RR:PP:QQ.**'
  ```

  rather than

  ```
  edit string 'YYYY-NN-DDBRR:PP:QQ.**'
  ```

- The formatting string ** represents the 100ths of a second field. Prior versions using a narrow field * would erroneously truncate the leading digits. This is corrected in this release, as the trailing digit is truncated.

4.1.17 Changes and Improvements to the Rdb Optimizer and Query Compiler

This release of Oracle Rdb introduces several new capabilities within the query compiler and the query optimizer. These changes fall generally under the title *query rewrite*, and allow the query compiler to present a simplified query for optimization and execution.

- CAST function elimination

  In most cases, CAST actions must be executed at runtime to convert from the source data type to that specified by the CAST function. However, in some cases, the Rdb query compiler can eliminate or replace the CAST function with a literal value during query compile. This saves CPU time as the action is performed just once rather than once per row processed.

  This replacement includes the following:

  – When CAST of DATE (ANSI), DATE (VMS) or TIMESTAMP data types is performed to a compatible type of DATE or TIMESTAMP, then in many cases the CAST operator is not required.

  – CAST of string literals to DATE (ANSI), DATE (VMS), TIME, TIMESTAMP and INTERVAL can be processed at compile time. For example, CAST('2013-1-1' AS DATE ANSI) is implicitly converted to a DATE literal DATE'2013-1-1'.

  – CAST of small integer values is now done by the compiler. For example, CAST(1 AS SMALLINT) can be performed at compile time.

  – CAST of fixed length (CHAR) literal strings to varying length strings (VARCHAR) is now processed by the compiler if the character set is the same and the target VARCHAR is long enough to hold the source string, as seen in the following example:

    ```
    CAST('TABLE' AS VARCHAR(31))
    ```

- Constant Folding

  Simple arithmetic expressions involving integer or floating point literals are evaluated by the query compiler. The overall effect is smaller executable code and some reduced CPU time for queries. FLOAT, REAL, and DOUBLE PRECISION values are combined to produce DOUBLE PRECISION results. Integer literals (with no fractional component) are combined to produce BIGINT results.

  The side effect is that some expressions may now return DOUBLE PRECISION or BIGINT results where in prior versions they produced smaller precision results. This should not affect applications which fetch values into different data types as Oracle Rdb will perform an implicit conversion.

  This optimization includes the following:

  * Addition (+)

  * Subtraction (-)

  * Multiplication (*)

  * Division (/)

    Note that division is not performed at compile time if the divisor is a literal zero (0). Operations which are coded to explicitly divide by zero are probably expected to produce an error at runtime. Although using the

SQL SIGNAL statement is now preferred, this technique has been used to terminate procedures when an incorrect input is encountered.

- Algebraic Rules

  Additive identity (zero) can be added to an expression without changing the value. The query compiler will eliminate the literal zero (0) from the expression.

  Multiply by zero will result in zero if the other operand is a not nullable expression. In this case, the expression will be replaced by zero.

  Multiplicative identity (one) can be multiplied by an expression without changing the value. The query compiler will eliminate the literal one (1) from the expression.

  The side effect is that some expressions may now return slightly different data types because the literal is no longer considered as part of the data type computation.

- Simple Predicate Elimination

  When predicates include comparison of simple expressions, then the query compiler will attempt to eliminate them from the query predicate. For example, WHERE ('A' = 'A') will be replaced by TRUE, WHERE (2 <> 2) will be replaced with FALSE, and so on.

- Not Nullable Aware

  The query compiler is now aware of which columns have a NOT NULL NOT DEFERRABLE constraint enabled. Additionally, this attribute is also implied from any PRIMARY KEY NOT DEFERRABLE constraints.

  Using this knowledge, the query compiler can reduce (prune) the query expression. This list defines the ways in which this can occur:

  * When IS NULL is applied to a not nullable column or expression, then this predicate is replaced with FALSE.

  * When IS NOT NULL is applied to a not nullable column or expression, then this predicate is replaced with TRUE.

  The side effect is that constraints for a table are now loaded for SELECT statements.

  This optimization can be disabled using the SET FLAGS statement, or the RDMS$SET_FLAGS logical name with the value NOREWRITE(IS_NULL). The default is REWRITE(IS_NULL).

- Replace comparisons with NULL

  Queries that erroneously compare value expressions with NULL will now be replaced with a simplified UNKNOWN value. For example, a query that uses WHERE EMPLOYEE_ID = NULL will never find matching rows, because the results of the comparison (equals, not equals, greater than, less than, and so on) are always UNKNOWN.

  This optimization can be disabled using the SET FLAGS statement, or the RDMS$SET_FLAGS logical name with the value NOREWRITE(UNKNOWN). The default is REWRITE(UNKNOWN).

- Predicate Pruning

The AND, OR and NOT operators can be simplified if the logical expressions have been reduced to TRUE, FALSE or UNKNOWN expressions. Depending on the operation, the Rdb query compiler might be able to eliminate the Boolean operator and part of the expression.

This optimization can be disabled using the SET FLAGS statement, or the RDMS$SET_FLAGS logical name with the value NOREWRITE(BOOLEANS). The default is REWRITE(BOOLEANS).

- CASE Expression Pruning

  The prior transformation will also be applied to the Boolean WHEN expressions of a conditional expression (CASE, DECODE, NULLIF, COALESCE, NVL, NVL2, SIGN, ABS, and so on).

  In some cases, the resulting conditional expression might resolve to an equivalent conditional expression with fewer branches (some WHEN ... THEN clauses being eliminated) or a simple expression with no conditional expression (all WHEN ... THEN clauses are eliminated).

- IN Operator Simplification

  The IN operator using a subquery looks similar to the EXISTS boolean expression but it differs in its handling of NULL values. If the query compiler knows that neither source field nor the value set contain NULL, then the EXISTS expression can replace the IN operator. The EXISTS expression generates a better query solution in almost all cases.

  This optimization can be disabled using the SET FLAGS statement, or the RDMS$SET_FLAGS logical name with the value NOREWRITE(IN_CLAUSE). The default is REWRITE(IN_CLAUSE).

In most cases, the results of these optimizations will be transparent to the application. However, database administrators that use SET FLAGS 'STRATEGY,DETAIL' will notice new notations in the displayed strategy.

The following examples show the types of likely results.

In this example, the logical expression (1 = 2) is replaced with FALSE, the logical expression (1 = 1) is replaced with TRUE and the predicate is reduced to just the IS NULL (aka MISSING) check.

```
SQL> select last_name
cont> from employees
cont> where ((1 = 1) and employee_id is null)
cont>        or
cont>       ((1 = 2) and employee_id = '00164');
Tables:
  0 = EMPLOYEES
Conjunct: MISSING (0.EMPLOYEE_ID)
Get     Retrieval sequentially of relation 0:EMPLOYEES
0 rows selected
```

If there existed a NOT NULL NOT DEFERRABLE constraint on the EMPLOYEE_ID column, the expression can be further reduced because the NOT NULL constraint means the IS NULL test is always FALSE.

```
SQL> alter table EMPLOYEES
cont>     alter column EMPLOYEE_ID
cont>         constraint NN_EMPLOYEE_ID
cont>             NOT NULL
cont>             NOT DEFERRABLE
cont> ;
SQL>
SQL> select last_name
cont> from employees
cont> where ((1 = 1) and employee_id is null)
cont>        or
cont>       ((1 = 2) and employee_id = '00164');
Tables:
  0 = EMPLOYEES
Conjunct: FALSE
Get     Retrieval sequentially of relation 0:EMPLOYEES
0 rows selected
SQL>
```

REWRITE Flag

The SET FLAGS statement and the RDMS$SET_FLAGS logical name can be used to enable or disable some of these rewrite actions. This flag primarily exists for Oracle to test the behavior of the query rewrite changes. It can be used by programmers to revert to pre-V7.3 behavior.

REWRITE enables each rewrite setting and NOREWRITE disables them. Additionally, keywords can be added to REWRITE and NOREWRITE to disable selective rewrite actions.

The following new keywords are added for this release of Oracle Rdb.

- BOOLEANS
- IN_CLAUSE
- IS_NULL
- UNKNOWN

4.1.18 Sorting Capabilities in Oracle Rdb

Oracle Rdb supports both the traditional OpenVMS SORT32 facility as well as simplified internal sort facility called QSORT.

QSORT

Use of QSORT preempts use of all other sorting algorithms. The QSORT algorithm is used if sorting is being done on a single key and if only a small amount of data is involved. The reason for this is that the other sorting algorithms, while handling larger data sets, have a certain amount of setup overhead which can be avoided in simple cases.

QSORT is used by default when:

- There is a single sort key.
- The number of rows to be sorted is 5000 or fewer. Note that this row limit can be adjusted.
- The buffer needed for caching the rows to be sorted is 409600 bytes (800 VMS pagelets) or less.
- The sort key is not floating point (REAL, FLOAT, or DOUBLE PRECISION).

Oracle Rdb supports two controls for the QSORT facility.

- RDMS$BIND_MAX_QSORT_COUNT

  This logical name controls the number of rows which will be buffered before resorting to the SORT32 interface.

  The default value is 5000 rows.

- RDMS$BIND_MAX_QSORT_BUFFER

  This logical acts as a governor to RDMS$BIND_MAX_QSORT_COUNT. If the rows being sorted are long, then the buffer size will reduce the count of candidate rows being used by QSORT. For example, even with the default values, a large row size (say 15304 bytes) will cause the actual limit on rows to be 26. This is displayed by the SORT display when SET FLAGS 'STRATEGY,SORT' is used.

```
SORT(18) SortId# 4, ---------- qsort used
  Records Input: 4      Record Limit: 26     LogRecLen Input: 15304
```

  The default value is 409600 bytes.

How to Disable QSORT

To disable QSORT because of either anomalous or undesirable performance, the user can define either of the following logicals to zero (0):

```
$ DEFINE RDMS$BIND_MAX_QSORT_BUFFER 0
$ DEFINE RDMS$BIND_MAX_QSORT_COUNT 0
```

How to Alter QSORT Usage

To change the usage of QSORT to evaluate behavior with other parameters, define new limits for the parameters as follows:

```
$ DEFINE RDMS$BIND_MAX_QSORT_BUFFER n
$ DEFINE RDMS$BIND_MAX_QSORT_COUNT m
```

The relationship between buffer size and row count is that the rows to be sorted are temporarily stored in the buffer until either it is full or the number of rows to be sorted exceeds the limit. For example, if the row to be sorted is 20 bytes long, 5000 of these rows would need 10,000 bytes which would easily fit into a 409600 byte buffer. However, if the rows to be sorted are 1,000 bytes long, only 409 of them can fit into a 409600 byte buffer and SORT32 will be invoked as soon as the sort algorithm is given row number 410.

4.1.19 RDM$BIND_MAX_DBR_COUNT Documentation Clarification

Bugs 1495227 and 3916606

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

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

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

The RDM$BIND_MAX_DBR_COUNT logical name defines the maximum number of database recovery (DBR) processes to be simultaneously invoked by the database monitor for each database during a "node failure" recovery. This logical name applies only to databases that do not have global buffers enabled. Databases that utilize global buffers have only one recovery process started at a time during a "node failure" recovery.

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

Per-Database Value

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

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

4.1.20 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 4–4 shows the TRANS_TPB table.

Table 4–4 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 4–5 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 4–5 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)	

4.1.21 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 4–6 shows the DATABASE table.

Table 4–6 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 4–7 shows the REQUEST_ACTUAL table.

Table 4–7 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	

(continued on next page)

Table 4–7 (Cont.) Columns for Table EPC$1_221_REQUEST_ACTUAL

Column Name	Data Type	Domain
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	

(continued on next page)

Table 4–7 (Cont.) Columns for Table EPC$1_221_REQUEST_ACTUAL

Column Name	Data Type	Domain
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	

(continued on next page)

Table 4–7 (Cont.) Columns for Table EPC$1_221_REQUEST_ACTUAL

Column Name	Data Type	Domain
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 4–8 shows the TRANSACTION table.

Table 4–8 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	

(continued on next page)

Table 4–8 (Cont.) Columns for Table EPC$1_221_TRANSACTION

Column Name	Data Type	Domain
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	

(continued on next page)

Table 4–8 (Cont.) Columns for Table EPC$1_221_TRANSACTION

Column Name	Data Type	Domain
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	

(continued on next page)

Table 4–8 (Cont.) Columns for Table EPC$1_221_TRANSACTION

Column Name	Data Type	Domain
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 4–9 shows the REQUEST_BLR table.

Table 4–9 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

4.2 RDO, RDBPRE and RDB$INTERPRET Features

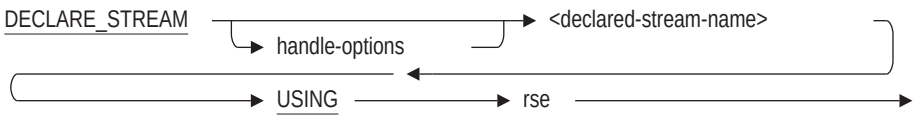
4.2.1 New Request Options for RDO, RDBPRE and RDB$INTERPRET

For release 7.0.1 of Oracle Rdb, two new keywords were added to the handle-options for the DECLARE_STREAM, the START_STREAM (undeclared format) and FOR loop statements. These changes have been made to RDBPRE, RDO and RDB$INTERPRET at the request of several RDO customers.

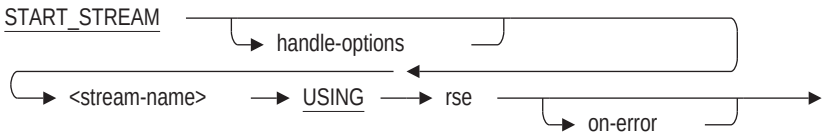
In prior releases, the handle-options could not be specified in interactive RDO or RDB$INTERPRET. This has changed in Rdb but these allowed options will be limited to MODIFY and PROTECTED keywords. For RDBPRE, all options listed will be supported. These option names were chosen to be existing keywords to avoid adding any new keywords to the RDO language.

The altered statements are shown below.

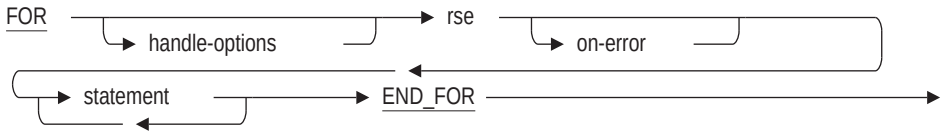
DECLARE_STREAM Format



START_STREAM Format

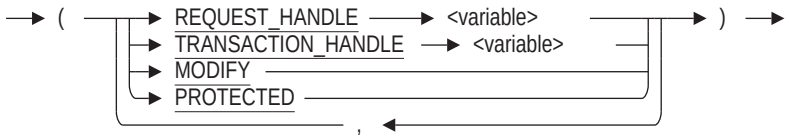


FOR Format



Each of these statements references the syntax for the HANDLE-OPTIONS which has been revised and is shown below.

handle-options =



The following options are available for HANDLE-OPTIONS:

- REQUEST_HANDLE specifies the request handle for this request. This option is only valid for RDBPRE and RDML applications. It cannot be used with RDB$INTERPRET, nor interactive RDO.
- TRANSACTION_HANDLE specifies the transaction handle under which this request executes. This option is only valid for RDBPRE and RDML applications. It cannot be used with RDB$INTERPRET, nor interactive RDO.
- MODIFY specifies that the application will modify all (or most) records fetched from the stream or for loop. This option can be used to improve application performance by avoiding lock promotion from SHARED READ for the FETCH to PROTECTED WRITE access for the nested MODIFY or ERASE statement. It can also reduce DEADLOCK occurrence because lock promotions are avoided. P> This option is valid for RDBPRE, RDB$INTERPRET, and interactive RDO. This option is not currently available for RDML.

  For example:

```
RDO>   FOR (MODIFY) E IN EMPLOYEES WITH E.EMPLOYEE_ID = "00164"
cont>    MODIFY E USING E.MIDDLE_INITIAL = "M"
cont>    END_MODIFY
cont>  END_FOR
```

  This FOR loop uses the MODIFY option to indicate that the nested MODIFY is an unconditional statement and so aggressive locking can be undertaken during the fetch of the record in the FOR loop.

- PROTECTED specifies that the application may modify records fetched by this stream by a separate and independent MODIFY statement. Therefore, this stream should be protected from interference (aka Halloween affect). The optimizer will select a snapshot of the rows and store them in a temporary relation for processing, rather than traversing indexes at the time of the FETCH statement. In some cases, this may result in poorer performance when the temporary relation is large and overflows from virtual memory to a temporary disk file, but the record stream will be protected from interference. The programmer is directed to the documentation for the Oracle Rdb logical names RDMS$BIND_WORK_VM and RDMS$BIND_WORK_FILE.

  This option is valid for RDBPRE, RDB$INTERPRET, and interactive RDO. This option is not currently available for RDML.

The following example creates a record stream in a BASIC program using Callable RDO:

```
RDMS_STATUS = RDB$INTERPRET ('INVOKE DATABASE PATHNAME "PERSONNEL"')

RDMS_STATUS = RDB$INTERPRET ('START_STREAM (PROTECTED) EMP USING ' + &
                             'E IN EMPLOYEES')

RDMS_STATUS = RDB$INTERPRET ('FETCH EMP')

DML_STRING = 'GET ' +                                          &
                 '!VAL = E.EMPLOYEE_ID;' +                     &
                 '!VAL = E.LAST_NAME;' +                       &
                 '!VAL = E.FIRST_NAME' +                       &
             'END_GET'

RDMS_STATUS = RDB$INTERPRET (DML_STRING, EMP_ID, LAST_NAME, FIRST_NAME)
```

In this case, the FETCH needs to be protected against MODIFY statements which execute in other parts of the application.

The problem was corrected in Oracle Rdb Release 7.0.1.

4.2.2 New Language Features for RDO and Rdb Precompiler

The following new language enhancements have been made to RDO, the Rdb Precompiler (RDBPRE), and the RDO Interpreter (RDB$INTERPRET).

- LIKE operator

  ```
  --> <value_expr> LIKE <value_expr> ->
  ```

  The rse WITH clause can now specify a LIKE relational operator, which is similar in action to the MATCHING operator. The LIKE operator returns TRUE if the second expression pattern matches the first value expression. LIKE is case sensitive. LIKE uses these special characters:

  - % Matches any string
  - _ Matches any character
  - \ an escape character. Use \\ to represent a single \, \% to represent a literal "%", and _ to represent a literal "_".

  This example is looking for any names starting with one character followed by an apostrophe.

  ```
  RDO> for e in employees
  cont>     with e.last_name like '_''%'
  cont>     print e.last_name
  cont> end_for
   LAST_NAME
   D'Amico
   O'Sullivan
  RDO>
  ```

- FIRST VIA ... FROM sub-query expression

  RDO includes a FIRST ... FROM sub-query expression. It returns the value from the matching row. However, if no rows are found, then the query will be aborted with a returned exception.

  The following example wishes to list each relation and its associated storage map (if it exists), and shows the reported RDB-E-FROM_NO_MATCH error.

```
RDO> for r in rdb$relations
cont>     with r.rdb$system_flag = 0
cont>     sorted by r.rdb$relation_name
cont>     print r.rdb$relation_name,
cont>       first sm.rdb$map_name from sm in rdb$storage_maps with
cont>             sm.rdb$relation_name = r.rdb$relation_name
cont> end_for
 RDB$RELATION_NAME                SM.RDB$MAP_NAME
 CANDIDATES                       CANDIDATES_MAP
%RDB-E-FROM_NO_MATCH, no record matched the RSE in a "from" expression
RDO>
```

RDO now supports an alternative to the FIRST ... FROM sub-query expression which modifies the behavior when no matching rows were selected by the sub-query. Adding the VIA keyword requests that a MISSING value be returned in such cases, and the query is no longer aborted.

```
RDO> for r in rdb$relations
cont>     with r.rdb$system_flag = 0
cont>     sorted by r.rdb$relation_name
cont>     print r.rdb$relation_name,
cont>       first via sm.rdb$map_name from sm in rdb$storage_maps with
cont>             sm.rdb$relation_name = r.rdb$relation_name
cont> end_for
 RDB$RELATION_NAME                SM.RDB$MAP_NAME
 CANDIDATES                       CANDIDATES_MAP
 COLLEGES                         COLLEGES_MAP
 CURRENT_INFO
 CURRENT_JOB
 CURRENT_SALARY
 DEGREES                          DEGREES_MAP
 DEPARTMENTS                      DEPARTMENTS_MAP
 EMPLOYEES                        EMPLOYEES_MAP
 EMPS
 JOBS                             JOBS_MAP
 JOB_HISTORY                      JOB_HISTORY_MAP
 RESUMES                          RESUMES_MAP
 SALARY_HISTORY                   SALARY_HISTORY_MAP
 WORK_STATUS                      WORK_STATUS_MAP
RDO>
```

- New special functions: RDO$CURRENT_USER, RDO$SESSION_USER and RDO$SYSTEM_USER

  These functions return the user identification of the current, session and system users. They can appear in any place that a field (aka column) can be used. These functions simplify view and trigger definitions created through RDO.

  Any view, computed by field, or trigger created by RDO but executed by a SQL session may return different values for each function. However, RDO sessions will typically return the same value from each function.

  This query uses the RDO$CURRENT_USER function to select the tables and views created by a user.

```
RDO> for r in rdb$relations
cont>     with r.rdb$relation_creator = rdo$current_user
cont>     print r.rdb$relation_name, r.rdb$created
cont> end_for
 RDB$RELATION_NAME                RDB$CREATED
 EMKP                             23-JUN-2014 12:53:26.28
 PICK_HISTORY_REC                 31-JUL-2015 15:11:47.46
 TEST_TABLE                        8-AUG-2014 08:21:25.96
 CUSTOMER_REC                      8-AUG-2014 08:17:16.34
 TAB1                              8-AUG-2014 08:17:17.88
 TAB2                              8-AUG-2014 08:17:17.88
 JOB_HIST                         15-AUG-2014 13:49:07.39
 SAL_HIST                         15-AUG-2014 13:49:07.39
 EMP_NAMES                        14-OCT-2014 21:18:49.03
 CAND_NAMES                       14-OCT-2014 21:18:49.03
 ACTION_CODES                     14-OCT-2014 21:18:49.23
   .
   .
   .
```

4.2.3 RDO Interface Now Supports Synonym References

Bug 20355063

This release of Oracle Rdb adds minimal support for synonyms to RDO and RDBPRE. In prior versions, a synonym to a table (or view) was not recognized by the RDO interfaces. For instance, an application built against table names which were subsequently renamed using SQL would no longer compile because the synonyms established by the RENAME TABLE or ALTER TABLE ... RENAME TO statements were not recognized by RDBPRE or RDO.

This support allows queries that reference table or view synonyms to be processed by the RDBPRE precompiler and RDO interactive utility. In addition, most SHOW commands in RDO will recognize a table or view synonym.

Data definition (DDL) commands, such as DROP RELATION or DEFINE CONSTRAINT, do not accept a synonym name as input. For such operations, Oracle recommends using the Interactive SQL interface.

This problem has been corrected in Oracle Rdb Release 7.3.2.0. Synonyms for tables and views created using any of the following statements are now recognized by RDO.

- RENAME TABLE ...
- RENAME VIEW ...
- ALTER TABLE ... RENAME TO ...
- ALTER VIEW ... RENAME TO ...
- CREATE SYNONYMS ... FOR TABLE ...
- CREATE SYNONYMS ... FOR VIEW ...

4.3 Address and Phone Number Correction for Documentation

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

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

4.4 Online Document Format and Ordering Information

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

The Oracle Rdb documentation in Adobe Acrobat format is available on the Oracle Rdb Documentation web page:

`http://www.oracle.com/technetwork/products/rdb/documentation/index.html`

Customers should contact their Oracle representative to purchase printed documentation.

5
Known Problems and Restrictions

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

5.1 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.

5.1.1 The Format of the RMU/ANALYZE/BINARY_OUTPUT Files Can Change

Bug 19261529

The /BINARY_OUTPUT qualifier used with the Oracle Rdb RMU/ANALYZE command specifies a binary output *.UNL file and/or a record definition *.RRD file to be created by RMU/ANALYZE to save statistical data created while analysing an Oracle Rdb database and to then load the statistical data into an Oracle Rdb database table. This data can then be used by a user-written management application or procedure. See the Oracle Rdb RMU REFERENCE MANUAL for more information on the /BINARY_OUTPUT qualifier used with the RMU/ANALYZE command.

The format of the binary and record definition files created by the RMU/ANALYZE/BINARY_OUTPUT command can change as changes are made to the statistical data produced by RMU/ANALYZE. Creaters and maintainers of user-written management applications or procedures dependent on the format of these files should be aware that Oracle Rdb reserves the right to add, remove or move fields in this record definition file for each major release of Rdb, which can require changes in these user-written management applications or procedures. The release notes for each major version of Oracle Rdb will document any changes in the format of these files. To make users aware of this possibility, the following warning comment has now been added to the record definition file.

```
DEFINE RECORD name
DESCRIPTION IS /* Oracle Rdb V7.3-130
                  Oracle reserves the right to add, remove or move fields in
                  this record definition file for each major release of Rdb */ .
```

The following example shows a database table named POS being created to hold RMU/ANALYZE/PLACEMENT statistics. Then an RMU/ANALYZE/PLACEMENT/BINARY_OUTPUT command is executed to save binary placement statistics, for the EMP_EMPLOYEE_ID index in the MF_PERSONNEL database, to the POS.UNL file and the record format for these statistics to the POS.RRD file. The RMU/LOAD command is then executed to load the binary placement statistic records from the POS.UNL file into the POS table using the record field format defined in the POS.RRD file.

```
$ sql
attach 'file mf_personnel';

create table pos (
        RMU$DATE                DATE VMS,
        RMU$INDEX_NAME          CHAR(32),
        RMU$RELATION_NAME       CHAR(32),
        RMU$PARTITION_NAME      CHAR(32),
        RMU$AREA_NAME           CHAR(32),
        RMU$LEVEL               INTEGER,
        RMU$FLAGS               INTEGER,
        RMU$COUNT               INTEGER,
        RMU$DUPLICATE_COUNT     INTEGER,
        RMU$DUPLICATE_MAP_COUNT INTEGER,
        RMU$KEY_COUNT           INTEGER,
        RMU$DUPLICATE_KEY_COUNT INTEGER,
        RMU$DATA_COUNT          INTEGER,
        RMU$DUPLICATE_DATA_COUNT INTEGER,
        RMU$TOTAL_KEY_PATH      INTEGER,
        RMU$TOTAL_PAGE_PATH     INTEGER,
        RMU$TOTAL_BUFFER_PATH   INTEGER,
        RMU$MAX_KEY_PATH        INTEGER,
        RMU$MAX_PAGE_PATH       INTEGER,
        RMU$MIN_BUF_PATH        INTEGER);
commit;
exit
$ rmu/analyze /placement -
   /binary_output=(file=pos.unl,record_definition=pos.rrd) -
   mf_personnel EMP_EMPLOYEE_ID
$ rmu/load/rms=(file=pos.rrd) mf_personnel POS pos.unl
%RMU-I-DATRECREAD,  1 data records read from input file.
%RMU-I-DATRECSTO,   1 data records stored 3-DEC-2014 10:58:04
$ type pos.rrd

  DEFINE RECORD RMU$ANALYZE_PLACEMENT
  DESCRIPTION IS /* Oracle Rdb V7.3-130
                    Oracle reserves the right to add, remove or move fields in
                   this record definition file for each major release of Rdb */ .

$
```

5.1.2 RMU /VERIFY /KEY_VALUES May Fail on Some Indices

In some cases, the RMU/VERIFY/KEY_VALUES functionality may be unable to perform key value verification failing with an %RDMS-F-OUTLINE_FAILED error. This happens when the generated SQL query and query outline can not be used to ensure index only retrieval from the specified index. This may occur with multi-segment HASHED (ORDERED or SCATTERED) indices, or multi-segment SORTED (and SORTED RANKED) indices with the REVERSE attribute.

The following example shows the reported error.

```
$ rmu/verify/nolog/index=T5_IDENT_IMAGE_NDX /key_values VERIFY_KEY_VALUES_DB
%RDMS-F-OUTLINE_FAILED, could not comply with mandatory query outline directives
%RMU-I-NOTREQVFY, not all requested verifications have been performed
$
```

This problem is a known limitation in this release of Oracle Rdb.

5.1.3 REPLACE Statement Fails With Primary Key Constraint Failure When Used on a View

The REPLACE statement does not show the correct semantics when used to insert into a view defined on a table with a PRIMARY KEY constraint.

The following example shows the problem.

```
SQL> create table SAMPLE
cont>     (ident integer
cont>     ,prod_name char(20)
cont>     ,primary key (ident) not deferrable
cont>     );
SQL>
SQL> create view SAMPLE_VIEW
cont>     (ident, prod_name)
cont>     as select * from SAMPLE
cont> ;
SQL>
SQL> insert into SAMPLE values (1, 'Ajax');
1 row inserted
SQL> insert into SAMPLE_VIEW values (2, 'Mr Clean');
1 row inserted
SQL>
SQL> replace into SAMPLE values (1, 'Borox');
1 row replaced
SQL> replace into SAMPLE_VIEW values (2, 'Mr. Clean');
%RDB-E-INTEG_FAIL, violation of constraint SAMPLE_PRIMARY_IDENT caused operation to fail
-RDB-F-ON_DB, on database RDB$DEFAULT_CONNECTION
SQL>
SQL> select * from SAMPLE order by ident;
       IDENT   PROD_NAME
           1   Borox
           2   Mr Clean
2 rows selected
SQL>
```

This will remain a restriction for this release. Only use REPLACE (or RMU/LOAD/REPLACE) on base tables.

5.1.4 Possible Incorrect Results When Using Partitioned Descending Indexes

Bug 6129797

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

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

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

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

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

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

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 while processing the index partitions.

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

5.1.5 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.

5.1.6 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.

5.1.7 External Routine Images Linked with PTHREAD$RTL

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

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

The shareable image RDB$NATCONN_FUNC73.EXE supplied with OCI Services for Oracle Rdb (part of SQL/Services) is one such shareable image that is linked with PTHREAD$RTL. Customer built applications that utilize External Routines from the RDB$NATCONN_FUNC73.EXE image must ensure that the main image is linked with PTHREAD$RTL. The external routines that a user may call that use functions from RDB$NATCONN_FUNC73.EXE include:

- TO_CHAR
- TO_NUMBER
- TO_DATE

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

5.1.8 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.

5.1.9 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.

5.1.10 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.

5.1.11 Optimization of Check Constraints

Bug 1448422

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

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

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

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

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

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

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

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

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

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

or:

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

In both cases, the retrieval strategy will look like this:

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

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

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

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

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

The strategy for a delete on T2 is now:

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

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

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

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

5.1.12 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.

5.1.13 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.

5.1.14 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.

5.1.15 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 5–1 Sort Memory Logicals

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

5.1.16 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.

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

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

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

This restriction does not apply to interactive or dynamic SQL.

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

```
1 MODULE sample_test
2 LANGUAGE C
3 PARAMETER COLONS
4
5 DECLARE ALIAS FILENAME 'mf_personnel'
6
7 PROCEDURE a (SQLCODE);
8 LOCK TABLE employees FOR EXCLUSIVE WRITE MODE;
%SQL-F-WISH_LIST, (1) Feature not yet implemented - LOCK TABLE requires compound
statement
9
10 PROCEDURE b (SQLCODE);
11 BEGIN
12 LOCK TABLE employees FOR EXCLUSIVE WRITE MODE;
13 END;
```

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

5.1.18 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.

5.1.19 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.

5.1.20 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
```

5.1.21 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.

5.1.22 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.

5.1.23 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.

5.1.24 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.

5.1.25 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.

5.1.26 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.

5.1.27 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.

5.1.28 Strict Partitioning May Scan Extra Partitions

When you use a WHERE clause with the less than (<) or greater than (>) operator and a value that is the same as the boundary value of a storage map, Oracle Rdb scans extra partitions. A boundary value is a value specified in the WITH LIMIT OF clause. The following example illustrates the behavior:

```
SQL> create table T1
cont>     (id integer
cont>     ,last_name char(12)
cont>     ,first_name char(12)
cont>     );
SQL> create storage map M for T1
cont>     partitioning not updatable
cont>     store using (id)
cont>         in EMPIDS_LOW with limit of (200)
cont>         in EMPIDS_MID with limit of (400)
cont>         otherwise in EMPIDS_OVER;
SQL> insert into T1 values (150,'Boney','MaryJean');
1 row inserted
SQL> insert into T1 values (350,'Morley','Steven');
1 row inserted
SQL> insert into T1 values (300,'Martinez','Nancy');
1 row inserted
SQL> insert into T1 values (450,'Gentile','Russ');
1 row inserted
SQL>
SQL> set flags 'EXECUTION(100),STRATEGY,DETAIL(2),INDEX_PARTITIONS';
SQL>
SQL> select * from T1 where ID > 400;
~S#0001
Tables:
  0 = T1
Conjunct: 0.ID > 400
Get     Retrieval sequentially of relation 0:T1         (partitioned scan#1)
~E#0001.1: Strict Partitioning using 2 areas
    partition 2 (larea=60) "EMPIDS_MID"
    partition 3 (larea=61) "EMPIDS_OVER" otherwise
          ID   LAST_NAME      FIRST_NAME
         450   Gentile        Russ
1 row selected
SQL>
```

In this example, partition 2 does not need to be scanned but is still accessed due to the structure of the generated key values. This does not affect the correctness of the result. Users can avoid the extra scan by using values other than the boundary values.

5.1.29 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.

5.1.30 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.

5.1.31 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.

5.1.32 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.

5.1.33 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 5–2 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

5.1.34 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.

5.1.35 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.