Oracle® Rdb for OpenVMS

Release Notes

Release 7.3.4

July 2023

ORACLE®

Oracle Rdb Release Notes, Release 7.3.4 for OpenVMS Industry Standard 64 for Integrity Servers and OpenVMS Alpha

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

Primary Author: Rdb Engineering and Documentation group

Contents

3 Software Errors Fixed in Oracle Rdb Release 7.3.3.2

4 Software Errors Fixed in Oracle Rdb Release 7.3.3.1

5 Enhancements and Changes Provided in Oracle Rdb Release 7.3.4

6 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.2

9 Known Problems and Restrictions

Examples

Tables

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb Release 7.3.4. The notes describe changed and enhanced features; upgrade and compatibility information; new and existing software problems and restrictions; and software and documentation corrections.

Intended Audience

This manual is intended for use by all Oracle Rdb users. Read this manual before you install, upgrade, or use Oracle Rdb Release 7.3.4.

Access to Oracle Support

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

Reader Comments

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title, release date, chapter, section, and page number (if available).

Please direct all comments, and corrections to this email address: **infordb_us@oracle.com**.

If you have problems with the software, please contact your local Oracle Support Services.

Online Document Format

You can view the Adobe Acrobat formatted documentation 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 can be reached from the Oracle Rdb main web page. At the bottom of the page are links to documentation for all of the Rdb Releases. Following is the link for the Oracle Rdb main page:

`https://www.oracle.com/database/technologies/related/rdb.html`

Optional Software

The following optional products may be referenced in these release notes:

- Oracle CDD/Repository
- Oracle Trace
- Oracle Replication Option for Rdb
- Oracle SQL/Services and OCI Services for Oracle Rdb
- Oracle SQL/Services Client API Kits
- Oracle ODBC Driver for Rdb
- Oracle JDBC for Rdb
- Oracle Rdb Developer Tools for Visual Studio
- Oracle Rdb Extension for SQL Developer
- Oracle Rdb Connectivity Manager

Document Structure

This manual consists of the following chapters:

Chapter 1	Describes how to install Oracle Rdb Release 7.3.4.
Chapter 2	Describes problems corrected in Oracle Rdb Release 7.3.4.
Chapter 3	Describes problems corrected in Oracle Rdb Release 7.3.3.2.
Chapter 4	Describes problems corrected in Oracle Rdb Release 7.3.3.1.
Chapter 5	Describes enhancements introduced in Oracle Rdb Release 7.3.4.
Chapter 8	Provides information not currently available in the Oracle Rdb documentation set.
Chapter 9	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb Release 7.3.4.

1

Installing Oracle Rdb Release 7.3.4

This software update is installed using the OpenVMS VMSINSTAL utility.

NOTE

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

1.1 Oracle Rdb on HPE OpenVMS Industry Standard 64

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

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

1.2 Requirements

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

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

- The installation requires approximately 400,000 blocks for OpenVMS Alpha systems.
- The installation requires approximately 600,000 blocks for OpenVMS I64 systems.

1.2.1 Ensure No Processes Have RDMSHRP Image Activated

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

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

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

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

```
$ @SYS$MANAGER:RMONSTOP73.COM
   .
   .
   .
$ @FIND_RDMSHRP73_PROC.COM

OpenVMS system analyzer

Process index: 016D   Name: FOO$SERVER  Extended PID: 2729F16D
 0240  7FEF4460 8384F300 $1$DGA2:[VMS$COMMON.SYSLIB]RDMSHRP73.EXE;222

$ STOP/IDENTIFICATION=2729F16D
```

1.3 Deprecated Features

1.3.1 Deprecated Support for the PL/I Compiler

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

1.4 Intel Itanium Processor 9700 “Kittson” Certified

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

1.5 Hardware Emulation Support

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

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

1.6 Maximum OpenVMS Version Check

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

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

1.7 Database Format Changed

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

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

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

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

1.8 Using Databases from Releases Earlier than V7.0

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

```
$ RMU/RESTORE/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 V7.3
%RMU-I-CVTCOMSUC, CONVERT committed for DEV1:[DOCUMENTATION.T]TESTING.RDB;1
to version V7.3
%RMU-W-USERECCOM, Use the RMU Recover command. The journals are not available.
```

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

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

Note

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

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

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

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

1.9 Invoking the VMSINSTAL Procedure

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

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

- To install the Oracle Rdb for OpenVMS I64 kit

  ```
  @SYS$UPDATE:VMSINSTAL RDBV73400IM073 device-name
  ```

- To install the Oracle Rdb for OpenVMS Alpha kit

  ```
  @SYS$UPDATE:VMSINSTAL RDBV73400AM073 device-name
  ```

device-name

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

1.10 Stopping the Installation

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

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

1.11 After Installing Oracle Rdb

This update provides a new Oracle TRACE facility definition for Oracle Rdb. Any Oracle TRACE selections that reference Oracle Rdb will need to be redefined to reflect the new facility version number for the updated Oracle Rdb facility definition, "RDBVMSV7.3".

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

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

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

   ```
   $ LIBRARY /TEXT /EXTRACT=RDBVMSV7.3 -
   _$ /OUT=RDBVMS.EPC$DEF SYS$SHARE:EPC$FACILITY.TLB
   ```

2. Insert the facility definition into the Oracle TRACE administration database.

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

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

1.12 VMS$MEM_RESIDENT_USER Rights Identifier Required

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

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

1.13 Installation, Configuration, Migration, Upgrade Suggestions

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

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

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

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

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

Table 1–1 Migration Suggestions

Case	You Wish To...	You should...
1	Add an Integrity server to an existing cluster of Alpha servers	1. Verify database(s) using RMU/VERIFY/ALL. 2. Backup database(s) using RMU/BACKUP. 3. Install Rdb 7.3 on Integrity and Alpha nodes. 4. Convert database(s) to the Rdb 7.3 structure level using RMU/CONVERT. 5. Verify database(s) again using RMU/VERIFY/ALL. 6. Backup database(s) using RMU/BACKUP. 7. Access database(s) from Alpha and Integrity directly by specifying database root file specification(s) in SQL ATTACH statements.

(continued on next page)

Table 1–1 (Cont.) Migration Suggestions

Case	You Wish To...	You should...
2	Add an Integrity server to an existing mixed cluster of VAX and Alpha nodes and access an Rdb database from all nodes. Disks used for the database are accessible from all nodes.	1. Verify database(s) using RMU/VERIFY/ALL. 2. Backup database(s) using RMU/BACKUP. 3. Install Rdb 7.3 on Integrity and Alpha nodes. 4. Convert database(s) to the Rdb 7.3 structure level using RMU/CONVERT. 5. Verify database(s) again using RMU/VERIFY/ALL. 6. Backup database(s) using RMU/BACKUP. 7. Access database(s) from Alpha and Integrity nodes directly by specifying database root file specification(s) in SQL ATTACH statements. 8. Access the database from VAX node(s) using the Rdb built-in network server (remote database) by specifying one of the Alpha or Integrity node names in SQL ATTACH statements. 9. After thorough testing, remove VAX nodes from the cluster.
3	Move database(s) to new disks and add an Integrity server to an existing cluster.	1. Use RMU/COPY with an options file to move the database files to the new disks. 2. Follow the steps for case 1 or case 2.
4	Continue to use Rdb primarily from VAX or Alpha nodes using earlier releases. Add an Integrity server for application testing purposes.	1. Install Rdb 7.3 on Integrity node. 2. Access existing database(s) from Integrity node by specifying one of the Alpha or VAX node names in the SQL ATTACH statements. 3. When testing is complete, follow the steps in case 1 or case 2.

(continued on next page)

Table 1–1 (Cont.) Migration Suggestions

Case	You Wish To...	You should...
5	Add an Integrity server to an existing cluster of Alpha servers or create a new cluster from an existing stand-alone Alpha server by adding one or more new Integrity servers.	1. Verify database(s) using RMU/VERIFY/ALL. 2. Backup database(s) using RMU/BACKUP. 3. Install Rdb 7.3 on Integrity and Alpha nodes. 4. Convert database(s) to the Rdb 7.3 structure level using RMU/CONVERT. 5. Verify database(s) again using RMU/VERIFY/ALL. 6. Backup database(s) using RMU/BACKUP. 7. Access database(s) from Alpha and Integrity directly by specifying database root file specification in the SQL ATTACH statements.
6	Create a new stand-alone Integrity Server system or cluster of Integrity Servers and move database(s) to the new environment.	1. Verify database(s) using RMU/VERIFY/ALL. 2. Install Rdb 7.3 on new system(s). 3. Back up database(s) on the existing cluster using RMU/BACKUP. 4. Copy backup file(s) to the new system (or, if using tape media, make the tapes available to the new system). 5. Restore database(s) on the new system using RMU/RESTORE specifying the location of each database file in an options file. 6. Verify the new database using RMU/VERIFY/ALL.

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

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

2

Software Errors Fixed in Oracle Rdb Release 7.3.4

This chapter describes software errors that are fixed by Oracle Rdb Release 7.3.4.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 More on the SYSTEM-F-ACCVIO in PSIINDEX2DIVIDESCR When Deleting Ranked Index Records

Bug 30457812

The problem discussed in the Oracle Rdb Release 7.3.3.2 Release Note "Unexpected SYSTEM-F-ACCVIO in PSIINDEX2DIVIDESCR When Deleting Ranked Index Records" was not fully resolved in the Oracle Rdb 7.3.3.2 release.

The following exception may still be raised during the deletion of records from a ranked index:

```
%SYSTEM-F-ACCVIO, access violation, reason mask=04,
virtual address=0000000003FBA000, PC=FFFFFFFF80002831, PS=00000009
Saved PC = 0000000080A4FA80 : RDMSHRP73\PSIINDEX2DIVIDESCR + 000002D0
Saved PC = 00000000809E2360 : RDMSHRP73\PSII2REMOVEBOTTOM + 000016D0
 . . .
```

In the scenarios investigated, the problem occurs only within the in-memory stuctures associated with the SORTED RANKED index processing code and does not result in permanent changes to the index on-disk structure.

Verification of the index prior to or after this problem occurs does not raise any index errors.

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

This problem has been corrected in Oracle Rdb Release 7.3.4.

2.1.2 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 referred 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 7.3.4. 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.

2.1.3 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 referred to as a "long verb" issue.

This problem has been corrected in Oracle Rdb Release 7.3.4. 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.

2.1.4 The SET FLAGS Option NOCOUNT_SCAN Erroneously Disabled MIN and MAX Optimized Index Lookups

On rare occasions, Oracle support may request that you use the Rdb database system debug flag "NOCOUNT_SCAN" to disable COUNT SCAN optimizations as a workaround for a problem you may be experiencing.

In prior releases of Oracle Rdb 7.3, the use of this flag also incorrectly suppressed the MIN/MAX key lookup optimizations carried out by the optimizer.

The following example shows this problem; selections no longer carry out "Min key lookup" or "Max key lookup" after disabling COUNT SCAN:

```
select max(employee_id) from employees;
Tables:
  0 = EMPLOYEES
Aggregate: 0:MAX (0.EMPLOYEE_ID) Q2
Index only retrieval of relation 0:EMPLOYEES
  Index name  EMP_EMPLOYEE_ID [0:0]      Max key lookup

 00471
1 row selected

! Now disable COUNT_SCAN
!
set flags 'nocount_scan';
```

```
select max(employee_id) from employees;
Tables:
  0 = EMPLOYEES
Aggregate: 0:MAX (0.EMPLOYEE_ID) Q2
Index only retrieval of relation 0:EMPLOYEES
  Index name  EMP_EMPLOYEE_ID [0:0]

 00471
1 row selected
SQL>
```

These problems have been corrected in Oracle Rdb Release 7.3.4. The NOCOUNT_SCAN keyword now only disables the COUNT (Index counts, Index counts lookup) and COUNT-DISTINCT (Index distinct counts, Index distinct lookup) optimizations when requested.

2.1.5 Unexpected ACCVIO and BUGCHECK Dump for Some Queries on Integrity systems

Bug 32931643

In prior releases of Oracle Rdb running on Integrity servers it was possible that large and complex queries or procedures might fail with an ACCVIO and have a bugcheck foot print similar to the following:

```
***** Exception at 000000008246D4C1 : symbol not found
%SYSTEM-F-ACCVIO, access violation, reason mask=04, virtual
address=0000000000200850, PC=FFFFFFFF8246D4C1, PS=0000000B
Saved PC = 0000000081189030 : RDMSHRP73\RDMS$$EXE_LEAF +
0000C170
Saved PC = 00000000810E5E70 : RDMSHRP73\RDMS$$EXE_NEXT +
000073C0
Saved PC = 00000000810E76A0 : RDMSHRP73\RDMS$$EXE_NEXT +
00008BF0
Saved PC = 00000000811C4B40 : RDMSHRP73\RDMS$$C_EXE_NEXT +
00000070
Saved PC = 000000008097FE40 : RDMSHRP73\RDMS_EXE_INTERP +
0000F3E0
Saved PC = 00000000811BDEF0 : RDMSHRP73\JSB_REQX + 00000120
Saved PC = 00000000811BCDD0 : RDMSHRP73\RDMS$TOP_START_
REQUEST + 00001A00
Saved PC = 0000000081CCA1A0 : Image RDMSHRP73 + 000001A0
```

A temporary workaround would be to disable the code optimization for the affected query or application. Please note that the affected applications will likely run more slowly since optimization is disabled. This workaround should be removed once this update is installed.

This problem has been corrected in Oracle Rdb Release 7.3.4. These large queries are correctly handled by Oracle Rdb.

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 an ALTER DATABASE statement to bugcheck if one of the storage area files was missing. For example, when using the ALTER DATABASE . . . DROP STORAGE AREA . . . CASCADE statement for an area that is referenced by an index or a table storage map the update of the storage map would result in an exception.

This problem has been corrected in Oracle Rdb Release 7.3.4. 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 When 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 7.3.4.

2.2.3 Unexpected NULL Returned From MOD and SQRT Functions

Bug 32038549

In prior releases of Oracle Rdb, the MOD and SQRT functions erroneously returned NULL once these functions accepted a NULL parameter value.

```
SQL> select * from x order by id;
          ID             I
           1           123
           2          NULL
           3           456
3 rows selected
SQL> select id,i,mod(i,100) from x order by id;
          ID             I
           1           123                      23
           2          NULL                    NULL
           3           456                    NULL
3 rows selected
SQL>
```

As a work around the MOD and SQRT definitions in SQL_FUNCTIONS73.SQL could be used. Ensure that the optional SQL functions are defined in the database using SYS$LIBRARY:SQL_FUNCTIONS73.SQL. In such cases these functions must be executed using a quoted name to avoid activating the builtin functions.

```
SQL> set quoting rules 'sql99';
SQL> select id,i,"MOD"(i,100) from x order by id;
          ID             I
           1           123    2.300000000000000E+001
           2          NULL                      NULL
           3           456    5.600000000000000E+001
3 rows selected
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.4.

2.2.4 Unexpected RDMS-E-UNQCONFEVAL When Using ALTER CONSTRAINT Statement

Bug 31914348

In prior releases of Oracle Rdb, attempts to use the ALTER CONSTRAINT statement to change a FOREIGN KEY constraint to NOT DEFERRABLE would result in an error. The following example shows this problem.

```
SQL> create table T1 (
cont>       C1
cont>           INTEGER,
cont>       C2
cont>           INTEGER,
cont>       constraint PK1
cont>           primary key (C1)
cont>           initially deferred deferrable);
SQL>
SQL> create table T2 (
cont>       C1
cont>           INTEGER,
cont>       constraint FK1
cont>           foreign key (C1)
cont>               references T1 (C1)
cont>           initially deferred deferrable);
SQL>
SQL> alter constraint PK1 not deferrable;
SQL>
SQL> alter constraint FK1 not deferrable;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-UNQCONFEVAL, the constraint "PK1" referenced by "FK1" has
    a conflicting evaluation time attribute
-RDMS-E-CONNOTCHG, constraint "FK1" has not been changed
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.4. Such an ALTER CONSTRAINT statement is legal and no longer produces an error.

When SET DIALECT 'SQL2011' or 'ORACLE LEVEL3' is in effect, the ANSI and ISO SQL Database Language rules apply and an attempt to alter the PRIMARY KEY or UNIQUE constraint referenced by a FOREIGN KEY constraint will result in an error. For other dialects a warning will be issued.

2.2.5 Unexpected Bugcheck When Query Contained Multiple COUNT(*) Aggregate Functions

Bug 31680491

Oracle Rdb detects when MIN (column), MAX (column), COUNT(*), COUNT(column) and COUNT(DISTINCT column) all reference the same indexed column and generates multiple optimized index lookups to resolve these aggregates.

When COUNT(*) is combined with a MAX, MIN or COUNT(column) it utilizes the index derived by those column references for the COUNT functionality.

In the problem case two (or more) COUNT(*) appear in the query triggering the multiple aggregates optimization. However, there is no target index and this bugcheck results.

```
SQL> select
cont>    count(*)-50  as discnt50
cont>    ,count(*)-10 as discnt10
cont>    ,count(*)    as fullcnt
cont> from employees;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USERS1:[TESTING]RDSBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
 address=0000000000000028, PC=FFFFFFFF806635D0, PS=0000001B
```

This problem has been corrected in Oracle Rdb Release 7.3.4.

2.2.6 Unexpected SQL-F-BUGCHK When Using EXEC SQL SAVEPOINT Statement

Bug 35469620

Attempts to use the SAVEPOINT statement in an EXEC SQL statement (SQL Precompiler) or within a PROCEDURE (SQL Module Language) resulted in a SQL-F-BUGCHK error.

```
$ sql$mod test_mod.sqlmod
%SQL-F-BUGCHK, There has been a fatal error.
    Please contact your Oracle support representative. gem$gendml - 2
```

This problem has been corrected in Oracle Rdb Release 7.3.4. Both the SQL Module Language and SQL Precompiler correctly handle the SAVEPOINT statement.

2.2.7 Incorrect Handling of the DEFAULT Clause for AUTOMATIC UPDATE AS Columns

In prior releases of Oracle Rdb the DEFAULT keyword could be used to assign a value to any generated column; IDENTITY, AUTOMATIC AS, or GENERATED AS column. However, if the column was defined by the AUTOMATIC UPDATE AS clause and also included the DEFAULT clause in the definition an incorrect value was returned for the DEFAULT keyword during the INSERT or REPLACE statements. Oracle Rdb erroneously used the computed expression for the column instead of the DEFAULT. For AUTOMATIC UPDATE AS the computed expression should only be stored during an UPDATE statement.

This problem has been corrected in Oracle Rdb Release 7.3.4. Oracle Rdb now correctly handles the DEFAULT keywords for AUTOMATIC UPDATE AS columns. If there is no DEFAULT defined for the column then the result will be to assign NULL to the column.

2.3 RMU Errors Fixed

2.3.1 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 7.3.4.

2.3.2 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 7.3.4 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.

2.3.3 Journal Backed Up While RMU/UNLOAD/AFTER/CONTINUOUS Extracting Journal

Bug 18601419

In rare circumstances it was possible for a journal backup to start executing a backup on the database's current journal at the same time that a journal extraction (RMU/UNLOAD/AFTER/CONTINUOUS) was starting on the same journal. That could sometimes result in the extraction process not extracting all of the contents of the journal. No error would be displayed by the extraction process and it would continue normally.

To avoid this issue, do not allow journal backups to run until the journal extraction process has completed its initialization.

This problem has been corrected in Oracle Rdb Release 7.3.4. RMU/UNLOAD/AFTER/CONTINUOUS has been changed to complete its initial journal checkpoint sequence before allowing journal backups to proceed.

2.3.4 Unexpected SYSTEM-F-ACCVIO When Using RMU Dump Audit for DACCESS

Bug 32288393

In prior releases of Oracle Rdb the RMU Dump Audit command may encounter an unhandled error and report a SYSTEM-F-ACCVIO exception when processing records in the AUDIT journal.

The following example shows the problem.

```
$ rmu/dump/audit -
    arg_database -
    RMU_AJ -
    /type=(DACCESS) -
    /format=LIST -
    /output=audit_dump.txt
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=0000000000000002,
    PC=FFFFFFFF804D8F80, PS=0000001B
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-F-FTL_RMU, Fatal error for RMU operation at 15-DEC-2020 09:43:45.04
$
```

This problem only affects the /FORMAT=LIST display (which is the default). As a workaround the qualifier /FORMAT=XML may be be used to display the audit records, or the data can be successfully unloaded using RMU/UNLOAD/AUDIT command.

This problem has been corrected in Oracle Rdb Release 7.3.4.

2.3.5 RMU Backup After_Journal Space_Check Now Supports Disk Bound Volume Sets

Bug 32320241

In prior releases of Oracle Rdb, the RMU Backup After_journal Space_check qualifier only checked the free space available on the first volume of a bound volume disk set.

The Space_Check qualifier is the default for the RMU Backup After_journal command, therefore, use Nospace_Check as a workaround if the target device of RMU Backup is a bound volume set.

This problem has been corrected in Oracle Rdb Release 7.3.4. RMU now checks the total free space available on all members of a bound volume set.

2.3.6 Bugchecks at SEQ$GET_SEQUENCE + 000001C0 After Incremental Restore

Bug 31826798

After restoring a database and applying an incremental backup, attempts to fetch a sequence number from the database failed with the following exception.

```
***** Exception at 0000000081E65140 : RDMSHRP73\SEQ$GET_SEQUENCE + 000001C0
%COSI-F-BUGCHECK, internal consistency failure
Saved PC = 000000008158DAA0 : RDMSHRP73\RDMS$$SEQUENCE_GET_NEXT + 000000C0
```

This problem would only occur when additional journals were added to the database after a full backup was performed and before an incremental backup was executed.

1. RMU/BACKUP
2. RMU/SET AFTER /RESERVE=
3. RMU/BACKUP/INCREMENTAL

Various data structures are stored in an Oracle Rdb database root (.RDB) file, including the sequence numbers for a database and the list of journal files. If any of those structures change in size then the contents of the root file must be shuffled to accommodate the new size. The offsets for each of those data structures are saved at the beginning of the root file.

In this situation, after new journals were added to the database, the sequence number structures were moved. The new offset for the sequence number structure was backed up in the incremental backup. However, when the database incremental backup was applied to the restored database, RMU did not update the database root file with the new offsets for the various root file data structures. Subsequent attempts to access those structures could fail since some structures might not be at the expected location. Using the wrong offset would cause unrelated data to be read which could lead to various failures.

To avoid this problem, always issue a full backup when changing the following values.

- Number of journals
- Number of storage areas
- Number of row caches
- Number of sequences

This problem has been corrected in Oracle Rdb Release 7.3.4. When RMU processes an incremental restore it now properly updates the data structure offsets in the database root file.

2.3.7 Average Values Could Be Zeroed on RMU Show Statistics Screens

It was reported that the "Summary IO Statistics", when displayed by RMU Show Statistics, would show the average values as zeroes. This occurred if the numeric values used to calculate the moving average values had became very large; such as when statistics have been collected over a long time period.

This problem has been fixed by increasing the numerical precision used to calculate these moving average values.

This problem has been corrected in Oracle Rdb release 7.3.4.

2.3.8 RMU Recover Fails With RMU-F-CANTSYNCTSNS Using Correct Journal

Bug 33906596

After restoring a database, RMU Recover would sometimes fail with the following error, even though the correct journal was specified in the command.

```
%RMU-F-BACKUPNOAIJ, After Image Journaling was enabled after
    the database was backed up or has since been disabled and reinitialized
-RMU-F-CANTSYNCTSNS, Last committed TSN 610 in the after image
    journal file exceeds last committed TSN 609 in the database root
```

When database prestarted transactions were enabled (the default), there was a small timing window where the database backup could be assigned a transaction sequence number (TSN) that is less than the TSN stored in the active after-image journal. Attempts to use that journal to recover the restored database would fail with RMU-F-CANTSYNCTSNS.

The following scenario could lead to this problem.

1. There is an active user in the database that has committed an update and is sitting idle. By default, that user has a prestarted transaction active, even if no explicit SET TRANSACTION was executed. The prestarted transaction is assigned a TSN at the time that the previous transaction commits.
2. An after-image backup is executed. The newly activated journal is assigned the next database TSN. Executing an RMU Dump AFTER_JOURNAL command on the journal open record displays this value as "Last Commit TSN is".
3. The idle user executes another transaction and commits. Its older TSN becomes the last committed TSN for the database.
4. An online database backup is performed. The backup will record the last committed database TSN. This is displayed as "Latest full backup transaction sequence number is" in the RMU Dump HEADER output.
5. A journal backup is performed for the journal that was active at the time of the database backup.
6. The database is restored.
7. An RMU Recover command is executed, attempting to apply the backup of the journal that was active at the time the database full backup was performed.

When an RMU Recover command is started, RMU verifies that the specified journal is applicable to the database. RMU compares the last commit TSN in the journal with the latest backup TSN in the restored database. If the database TSN is greater than or equal to the journal's TSN then the journal is applied. Otherwise, an RMU-F-CANTSYNCTSNS error is issued. In the scenario described above, due to an old, prestarted transaction sequence number, this test would fail, even though the journal should have been applicable.

To workaround this problem, provide the prior AIJ backup as well as the expected journal to the RMU Recover command. In the following example, the expected journal is AIJ_BACKUP_1. For the command to succeed, the prior journal, AIJ_BACKUP_0 is specified along with AIJ_BACKUP_1.

```
$ RMU/RECOVER/LOG AIJ_BACKUP_0, AIJ_BACKUP_1
```

This problem will not occur if prestarted transactions are disabled.

```
SQL> ALTER DATABASE
cont>   FILENAME MF_PERSONNEL
cont>   PRESTARTED TRANSACTIONS ARE DISABLED
cont> ;
```

Note that disabling prestarted transactions will likely have a noticeable impact on performance in high transaction rate applications.

This problem has been corrected in Oracle Rdb Release 7.3.4. Now, when a quiet-point after-image journal backup is executed, all idle processes that have a prestarted transaction active are forced to obtain a new TSN when the next transaction is explicitly started.

2.3.9 RMU Verify Validate FLOAT Did Not Correctly Scan The Target Tables

Bug 35220492

RMU Verify Validate FLOAT was not processing the table completely and therefore not displaying any found bad floating point values.

The following example shows the expected output when bad values are found.

```
$       RMU/VERIFY/VALIDATE=FLOAT/LOG TESTING_DB
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-DBBOUND,   bound to database
    "USERS1:[TESTING.VALIDATE_FLOAT]TESTING_DB.RDB;1"
%RMU-I-OPENAREA,  opened storage area
    USERS1:[TESTING.VALIDATE_FLOAT]TESTING_DB.RDB;1 for protected retrieval
%RMU-I-BGNAIPVER, beginning AIP pages verification
%RMU-I-ENDAIPVER, completed AIP pages verification
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-I-ENDABMSPM, completed ABM pages verification
%RMU-I-CLOSAREAS, releasing protected retrieval lock on all storage areas
%RMU-I-BGNVALDAT,     Beginning of table data validation 28-MAR-2023 15:26:21.01
%RMU-I-TABCOLVLD, Scanning table FLOATS_TAB which has 2 floating point columns.
%RMU-I-INVDATF,     Invalid F floating data 01438000 in field F_FLOAT
    of table FLOATS_TAB at dbkey 57:498:2
%RMU-I-INVDATG,     Invalid G floating data 0000000001438000 in field G_FLOAT
    of table FLOATS_TAB at dbkey 57:498:2
%RMU-I-ENDVALDAT,     End of data validation at 28-MAR-2023 15:26:21.04
%RMU-S-ENDVERIFY, elapsed time for verification :    0 00:00:00.13
$
```

The provided DBKEY can be used to update the rows such as shown in this example.

```
SQL> select * from FLOATS_TAB where dbkey = _dbkey'57:498:2';
     ROW_NUM          F_FLOAT                   G_FLOAT
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-COSI-F-FLTINV, invalid floating conversion
SQL> update FLOATS_TAB set F_FLOAT = null, G_FLOAT = null
cont> where dbkey = _dbkey'57:498:2';
           3
1 row updated
SQL> select * from FLOATS_TAB;
     ROW_NUM          F_FLOAT                   G_FLOAT
           1    1.0000000E+00    1.000000000000000E+000
           2    2.0000000E+00    2.000000000000000E+000
           3            NULL                      NULL
3 rows selected
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.4.

2.3.10 SERVER_TRANSPORT Option to RMU Backup Parallel not Working as Documented

Bug 32929697

The Oracle Rdb RMU Reference Manual (and DCL HELP) describes the option SERVER_TRANSPORT as allowing the choice between DECNET or TCPIP. Unfortunately, this option is accepted but does not change the default setting.

By default when RMU Backup Parallel is executed it will attempt to first use DECnet transport and then re-try using TCP/IP upon failure. If /PARALLEL=(. . . , SERVER_TRANSPORT=TCPIP, . . .) is used then the fallback from a failed DECnet attempt will be avoided. The failure might occur because DECnet isn't installed, not active, or the current user does not have DECnet proxy definitions for the chosen cluster nodes. See the OpenVMS HELP for the AUTHORIZE utility ADD /PROXY command.

As an alternative to using the RMU Backup Parallel SERVER_TRANSPORT option you may use either the logical name RMU_NETWORK_TRANSPORT_TYPE or SQL_NETWORK_TRANSPORT_TYPE. Either logical name must be defined prior to running the RMU Backup Parallel Execute command or the RMU Backup Plan Execute command. These logical names expect an uppercase value for DECNET or TCPIP.

```
$ define /user RMU_NETWORK_TRANSPORT_TYPE "TCPIP"
$ rmu /backup -
    /parallel=(executor_count=3,nodes=(ALPHA,BETA,GAMMA)) -
    /execute -
    ...etc... -
    ADMIN_DB -
    ADMIN_BCK_JULY010
```

The setting for the SERVER_TRANSPORT option is now written to the generated Plan file when the List_Plan qualifier is used. However, if SERVER_TRANSPORT is not specified in the Parallel qualifier options then no SERVER_TRANSPORT setting is written to the Plan file. Therefore, if you plan to use either logical name then those logical names need to be defined prior to the RMU Backup . . . Execute command.

This problem has been corrected in Oracle Rdb Release V7.3.4. RMU Backup now correctly supports the RMU Backup Parallel SERVER_TRANSPORT option, and writes it to the plan file when using the List_Plan qualifier. The translation of the RMU_NETWORK_TRANSPORT_TYPE and SQL_NETWORK_TRANSPORT_TYPE logical names will only be used if the option SERVER_TRANSPORT is not specified.

2.4 LogMiner Errors Fixed

2.4.1 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 7.3.4.

3

Software Errors Fixed in Oracle Rdb Release 7.3.3.2

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

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 RCS Bugchecks in DIO$LACB_AIP_ENT_REFRESH After Drop Table or Index

Bugs 4738878, 20692223

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

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

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

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

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

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

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

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

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

This problem has been corrected in Oracle Rdb Release 7.3.3.2.

3.1.2 Unexpected SYSTEM-F-ACCVIO in PSIINDEX2DIVIDESCR When Deleting Ranked Index Records

Bugs 15855499, 11656062 and 30457812

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

```
%SYSTEM-F-ACCVIO, access violation, reason mask=04,
virtual address=0000000003FBA000, PC=FFFFFFFF80002831, PS=00000009
Saved PC = 0000000080A4FA80 : RDMSHRP73\PSIINDEX2DIVIDESCR + 000002D0
Saved PC = 00000000809E2360 : RDMSHRP73\PSII2REMOVEBOTTOM + 000016D0
...
```

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

This problem has been corrected in Oracle Rdb Release 7.3.3.2.

3.2 SQL Errors Fixed

3.2.1 Unexpected Routine Call Failure Reporting RDB-E-OBSOLETE_METADATA

Bug 29610611

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

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

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

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

or this error

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

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

This problem has been corrected in Oracle Rdb Release 7.3.3.2.

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

Bug 8941047

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

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

```
SQL> declare jq3
cont>     read only table cursor
cont>     for select resume res from resumes
cont>  ;
SQL>
SQL> declare jq4
cont>     read only scroll list cursor
cont>     for select resume
cont>         where current of jq3;
%SQL-F-NOTINTBLCUR, Column, RESUME, not specified in table cursor, JQ3
SQL>
SQL> declare jq6
cont>     read only scroll list cursor
cont>     for select res
cont>         where current of jq3;
%SQL-F-FLDNOTCRS, Column RES was not found in the tables in current scope
```

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

This problem has been corrected in Oracle Rdb Release 7.3.3.2.

3.2.3 Unexpected Value in SQLCHRONO_SCALE When SQLDA2 is Reused

Bug 30033543

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

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

3.2.4 Possible Wrong Result Returned From Aggregate Subselect Clauses

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

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

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

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

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

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

This problem has been corrected in Oracle Rdb Release 7.3.3.2.

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

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

Note

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

3.2.5 Unexpected SYSTEM-F-ACCVIO When Processing Complex Query

Bug 30382086

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

The following example shows an example of such a query.

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

This problem has been corrected in Oracle Rdb Release 7.3.3.2.

3.2.6 Unexpected Bugcheck When Trigger Executes in Called Stored Procedure

Bug 30520710

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

```
***** Exception at 00000000813C7870 : RDMSHRP73\RDMS$$EXECUTE_
ETRG + 000005D0
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=
00000000000
Saved PC = 000000008141A4F0 : RDMSHRP73\RDMS$$EXE_ACTION +
0000A490
Saved PC = 0000000081507020 : RDMSHRP73\RDMS$$C_EXE_ACTION +
00000070
Saved PC = 0000000080CC2730 : RDMSHRP73\RDMS_EXE_INTERP +
0000FD10
Saved PC = 00000000815005F0 : RDMSHRP73\JSB_REQX + 00000220
Saved PC = 0000000081443610 : RDMSHRP73\RDMS$TOP_START_SEND_
RECEIVE + 00002E90
```

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

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

3.3 RMU Errors Fixed

3.3.1 Unexpected Ordering of Fields in Extract Records From RMU Extract

Bug 29869005

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

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

The following example shows creation of the new column ordering.

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

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

SQL>
```

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

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

/* Table: SAMPLE (null terminated)
*/

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

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

3.3.2 Unexpected Corrupted Files From NFS (TCPIP) Mounted Device

Bug 30100614

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Bug 29844726

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

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

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

Here is the expected output from RMU Extract:

```
** Table: TEST_SCALED_INT

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

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

4

Software Errors Fixed in Oracle Rdb Release 7.3.3.1

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

4.1 Software Errors Fixed That Apply to All Interfaces

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

Bug 28604088

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

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

```
create database filename 'testdata';

create table T1 (
    JOIN_ID
        INTEGER,
    IDENT_ID
        INTEGER);

create table T2 (
    JOIN_ID
        INTEGER,
    IDENT_ID
        INTEGER);

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

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

create index T2_JOIN_NDX on T2 (JOIN_ID);

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

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

commit work;
```

```
! Now execute the following query
!
SELECT * from T1 t1 where ident_id IN (
select b.ident_id
  from T1 b
       inner join T2 v on ( b.join_id = v.join_id )
 where b.join_id = 123456 );
Tables:
  0 = T1
  1 = T1
  2 = T2
Conjunct: <agg0> <> 0
Match    (Agg Outer Join)  Q1
  Outer loop
  Match_Key:0.IDENT_ID
    Sort: 0.IDENT_ID(a)
    Get Retrieval sequentially of relation 0:T1
  Inner loop
  Match_Key:1.IDENT_ID
    Aggregate: 0:COUNT-ANY (<subselect>) Q2
    Cross block of 2 entries  Q2
      Cross block entry 1
        Leaf#01 Sorted 1:T1 Card=20000
          Bool: 1.JOIN_ID = 123456
          FgrNdx  T1_IDENT_NDX [0:0] Fan=46
          BgrNdx1 T1_JOIN_NDX [1:1] Fan=46
            Keys: 1.JOIN_ID = 123456
      Cross block entry 2
        Index only retrieval of relation 2:T2
          Index name  T2_JOIN_NDX [1:1]
            Keys: 1.JOIN_ID = 2.JOIN_ID
0 rows selected
```

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

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

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

```
SELECT * from T1 t1 where EXISTS (
select b.ident_id
  from T1 b
       inner join T2 v on ( b.join_id = v.join_id )
 where b.join_id = 123456 and
       b.ident_id = t1.ident_id);
Tables:
  0 = T1
  1 = T1
  2 = T2
Conjunct: <agg0> <> 0
Match  Q1
  Outer loop      (zig-zag)
  Match_Key:0.IDENT_ID
  Index_Key:IDENT_ID
    Get     Retrieval by index of relation 0:T1
      Index name  T1_IDENT_NDX [0:0]
  Inner loop
  Match_Key:1.IDENT_ID
    Aggregate: 0:COUNT-ANY (<subselect>) Q2
    Cross block of 2 entries  Q2
      Cross block entry 1
        Leaf#01 Sorted 1:T1 Card=20000
          Bool: 1.JOIN_ID = 123456
          FgrNdx  T1_IDENT_NDX [0:0] Fan=46
          BgrNdx1 T1_JOIN_NDX [1:1] Fan=46
            Keys: 1.JOIN_ID = 123456
      Cross block entry 2
        Index only retrieval of relation 2:T2
          Index name  T2_JOIN_NDX [1:1]
            Keys: 1.JOIN_ID = 2.JOIN_ID
0 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.3.3.1.

4.1.2 Ranked Index Bugcheck at PSIINDEX2JOINSCR

Bug 28854908

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

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

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

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

This problem has been corrected in Oracle Rdb Release 7.3.3.1.

4.2 SQL Errors Fixed

4.2.1 Unexpected Action When CREATE Routine Was Missing LANGUAGE Clause

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

these statements as DECLARE FUNCTION or DECLARE PROCEDURE statements.

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

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

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

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

This problem has been corrected in Oracle Rdb Release 7.3.3.1.

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

Bug 6502369

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

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

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

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

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

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

On the default alias
Transaction characteristics:
    Read Write

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

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

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

4.2.3 Unexpected Routine Call Failure Reporting RDB-E-OBSOLETE_METADATA

Bug 28255803

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

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

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

The following example shows the reported error.

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

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

Databases without SYNONYMS enabled are not affected by this problem.

This problem has been corrected in Oracle Rdb Release 7.3.3.1.

4.2.4 Unexpected RDB-E-INVALID_BLR When Defining a Constraint

Bug 4461005

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

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

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

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

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

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

This problem has been corrected in Oracle Rdb Release 7.3.3.1.

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

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

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

```
SQL> select count(*)
cont>  from EMPLOYEES E
cont>  where not
cont>     ( E.EMPLOYEE_ID is not null
cont>       and E.EMPLOYEE_ID in (select EMPLOYEE_ID from SUPERVISORS)
cont>     )
cont> ;
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (*) Q2
Conjunct: FALSE
Get     Retrieval sequentially of relation 0:EMPLOYEES

                    0
1 row selected
SQL>
```

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

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

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

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

This problem has been corrected in Oracle Rdb Release 7.3.3.1.

4.2.6 Unexpected Sequence Creation by DECLARE LOCAL TEMPORARY TABLE Statement

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

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

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

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

  This error is unexpected.

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

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

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

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

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

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

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

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

This problem has been corrected in Oracle Rdb Release 7.3.3.1. In this release of Oracle Rdb, the DECLARE LOCAL TEMPORARY TABLE statement will materialize a local (session) sequence matching the characteristics of the IDENTITY or GENERATED ... AS IDENTITY definition. Upon DISCONNECT, the state of the sequence will be discarded and it will be reloaded for the next connection. Each temporary sequence is private to the connection.

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

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

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

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

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

Additionally, DECLARE LOCAL TEMPORARY TABLE supports the enhanced LIKE table syntax. See Section 7.1.12 for more details.

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

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

4.3 RMU Errors Fixed

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

Bug 4461005

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

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

The following example shows the problem with the missing parentheses.

```
$ rmu/extract/item=view/option=(match=v00,noheader,filename) sql$database
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename PERSONNEL';
create view V00
    (EMPLOYEE_ID,
     START_DATE,
     END_DATE,
     TABLE_ABBREV) as
    select C2.EMPLOYEE_ID, C2.JOB_START, C2.JOB_END, 'JH'
        from JOB_HISTORY C2
        order by C2.EMPLOYEE_ID asc
    union all
    select C3.EMPLOYEE_ID, C3.SALARY_START, C3.SALARY_END, 'SH'
        from SALARY_HISTORY C3
        order by C3.EMPLOYEE_ID asc;
```

```
commit work;
$
```

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

```
$ rmu/extract/item=view/option=(match=v00,noheader,filename)-
    /output=q.sql -
    sql$database
$ sql$ @q
SQL> set language ENGLISH;
SQL> set default date format 'SQL92';
SQL> set quoting rules 'SQL92';
SQL> set date format DATE 001, TIME 001;
SQL> attach 'filename PERSONNEL';
SQL> create view V00
cont>     (EMPLOYEE_ID,
cont>      START_DATE,
cont>      END_DATE,
cont>      TABLE_ABBREV) as
cont>     (select C2.EMPLOYEE_ID, C2.JOB_START, C2.JOB_END, 'JH'
cont>         from JOB_HISTORY C2
cont>         order by C2.EMPLOYEE_ID asc)
cont>     union all
cont>     (select C3.EMPLOYEE_ID, C3.SALARY_START, C3.SALARY_END, 'SH'
cont>         from SALARY_HISTORY C3
cont>         order by C3.EMPLOYEE_ID asc);
SQL>
SQL> commit work;
$
```

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

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

The following example shows that RMU now displays this information.

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

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

This problem has been corrected in Oracle Rdb Release 7.3.3.1.

4.4 RMU Show Statistics Errors Fixed

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

Bug 28430946

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

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

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

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

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

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

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

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


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

record marked                   0        0        0.0            0           0.0
record fetched                  0        0        0.0            0           0.0
    fragmented                  0        0        0.0            0           0.0
record stored                   0        0        0.0            0           0.0
   fragmented                   0        0        0.0            0           0.0
pages checked                   0        0        0.0            0           0.0
 saved IO                       0        0        0.0            0           0.0
 discarded                      0        0        0.0            0           0.0
record erased                   0        0        0.0            0           0.0
 fragmented                     0        0        0.0            0           0.0
sequential scan                 0        0        0.0            0           0.0
 record fetched                 0        0        0.0            0           0.0


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

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

$set terminal/page=29
$rmu/show statistics mf_personnel
Node: TSTNOD (1/1/16)                        Oracle Rdb V7.3-300 Perf. Monitor
Rate: 3.00 Seconds                                Logical Area Statistics
Page: 1 of 1                          DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
................................................................................
                                                 Table CANDIDATES in RDB$SYSTEM


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

record marked                  0        0        0.0            0           0.0
record fetched                 0        0        0.0            0           0.0
    fragmented                 0        0        0.0            0           0.0
record stored                  0        0        0.0            0           0.0
   fragmented                  0        0        0.0            0           0.0
pages checked                  0        0        0.0            0           0.0
 saved IO                      0        0        0.0            0           0.0
 discarded                     0        0        0.0            0           0.0
record erased                  0        0        0.0            0           0.0
 fragmented                    0        0        0.0            0           0.0
sequential scan                0        0        0.0            0           0.0
 record fetched                0        0        0.0            0           0.0


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


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


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

record marked                  0        0        0.0            0           0.0
record fetched                 0        0        0.0            0           0.0
    fragmented                 0        0        0.0            0           0.0
record stored                  0        0        0.0            0           0.0
   fragmented                  0        0        0.0            0           0.0
pages checked                  0        0        0.0            0           0.0
 saved IO                      0        0        0.0            0           0.0
 discarded                     0        0        0.0            0           0.0
record erased                  0        0        0.0            0           0.0
 fragmented                    0        0        0.0            0           0.0
A. record marked               0        0        0.0            0           0.0
B. record fetched              0        0        0.0            0           0.0
C.     fragmented
D. record stored
E.    fragmented
F. pages checked
G.  saved IO      ..........................................................
H.  discarded     letter> to select logical area statistics, <control-Z> to cancel
```

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

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

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


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

record marked                  0        0        0.0            0           0.0
record fetched                 0        0        0.0            0           0.0
    fragmented                 0        0        0.0            0           0.0
record stored                  0        0        0.0            0           0.0
   fragmented                  0        0        0.0            0           0.0
pages checked                  0        0        0.0            0           0.0
 saved IO                      0        0        0.0            0           0.0
 discarded                     0        0        0.0            0           0.0
record erased                  0        0        0.0            0           0.0
 fragmented                    0        0        0.0            0           0.0
sequential scan                0        0        0.0            0           0.0
 record fetched                0        0        0.0            0           0.0


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


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


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

A. record marked               0        0        0.0            0           0.0
B. record fetched              0        0        0.0            0           0.0
C.     fragmented              0        0        0.0            0           0.0
D. record stored               0        0        0.0            0           0.0
E.    fragmented               0        0        0.0            0           0.0
F. pages checked               0        0        0.0            0           0.0
G.  saved IO                   0        0        0.0            0           0.0
H.  discarded                  0        0        0.0            0           0.0
I. record erased               0        0        0.0            0           0.0
J.  fragmented                 0        0        0.0            0           0.0
K. sequential scan             0        0        0.0            0           0.0
L.  record fetched             0        0        0.0            0           0.0
M. ALL

.....................................................................................
```

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

$set terminal/page=29
$rmu/show statistics mf_personnel
Node: TSTNOD (1/1/16)                          Oracle Rdb V7.3-310 Perf. Monitor
Rate: 3.00 Seconds                                 Logical Area Statistics
Page: 1 of 1                          DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
.............................................................................
                                                 Table CANDIDATES in RDB$SYSTEM


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

record marked                  0        0        0.0            0           0.0
record fetched                 0        0        0.0            0           0.0
    fragmented                 0        0        0.0            0           0.0
record stored                  0        0        0.0            0           0.0
   fragmented                  0        0        0.0            0           0.0
pages checked                  0        0        0.0            0           0.0
 saved IO                      0        0        0.0            0           0.0
 discarded                     0        0        0.0            0           0.0
record erased                  0        0        0.0            0           0.0
 fragmented                    0        0        0.0            0           0.0
sequential scan                0        0        0.0            0           0.0
 record fetched                0        0        0.0            0           0.0

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


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


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

A. record marked               0        0        0.0            0           0.0
B. record fetched              0        0        0.0            0           0.0
C.     fragmented              0        0        0.0            0           0.0
D. record stored               0        0        0.0            0           0.0
E.    fragmented               0        0        0.0            0           0.0
F. pages checked               0        0        0.0            0           0.0
G.  saved IO                   0        0        0.0            0           0.0
H.  discarded                  0        0        0.0            0           0.0
I. record erased               0        0        0.0            0           0.0
J.  fragmented                 0        0        0.0            0           0.0
K. sequential scan             0        0        0.0            0           0.0
L.  record fetched             0        0        0.0            0           0.0
M. ALL


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

This problem has been corrected in Oracle Rdb Release 7.3.3.1.

4.5 Oracle Trace Errors Fixed

4.5.1 Oracle TRACE Data Not Being Collected for All Database Connections

Bug 9728436

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

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

5

Enhancements and Changes Provided in Oracle Rdb Release 7.3.4

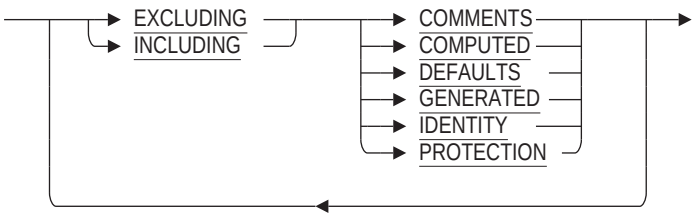
5.1 Enhancements and Changes Provided in Oracle Rdb Release 7.3.4

5.1.1 Enhanced LIKE Table Support in CREATE TABLE Statement

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

Syntax

like-attributes =



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

5.1.2 Some Aggregate Functions Now Inherit Source Column EDIT STRING

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

- MAX, MEDIAN, MIN, FIRST_VALUE, LAST_VALUE

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

- CAST

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

The following example shows the EDIT STRING being used.

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

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

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

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

5.1.3 New Summary_Only Qualifier to RMU Dump Audit Command

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

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

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

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

5.1.4 New Option=GENERATED Added to RMU Extract Command

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

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

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

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

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

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

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

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

Bug 31711278

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

- /EDIT_FILENAME

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

  This qualifier replaces any EDIT_FILENAME defined for the database.

- /NOEDIT_FILENAME

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

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

- Neither /EDIT_FILENAME nor /NOEDIT_FILENAME was used.

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

6

Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.2

6.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.2

6.1.1 New PCSI Support for Rdb Kit Installation and Deinstallation

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

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

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

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

File lookup pass starting ...

Portion done: 0%
...100%

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

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

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

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

1 item found

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

```
1 item found

$ PRODUCT SHOW PRODUCT RDB7*
---------------------------------- ----------- ---------
PRODUCT                            KIT TYPE    STATE
---------------------------------- ----------- ---------
ORCL I64VMS RDB74 V7.3-320         Transition  Installed
---------------------------------- ----------- ---------

1 item found
```

The following lines will now be displayed during the deinstallation of Oracle Rdb, showing that the removal of the release has been registered in the PCSI database. Deinstallation is performed by executing the DCL procedure 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 RDB73 V7.3-320            Transition (registration)

The following product will be removed from destination:
    ORCL I64VMS RDB73 V7.3-320            DISK$CLYPPR84_2:[VMS$COMMON.]

Portion done: 0%...100%

The following product has been removed:
    ORCL I64VMS RDB74 V7.3-320            Transition (registration)
```

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

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

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

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

6.1.2 Updated Support for RDMS$BIND_CODE_OPTIMIZATION

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

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

6.1.3 DUPLICATES ARE NOT ALLOWED Clause Added to ALTER INDEX Statement

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

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

Note

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

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

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

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

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

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

The new Oracle Rdb RMU SET DATABASE command ACCESS=[UN]RESTRICTED qualifier sets or clears the restricted database access flag in the database root file if the database is currently open or closed. If the restricted database access flag in the database root file is set, SQL users must have DBADM privilege to attach to an Rdb database. Previously, the restricted database access flag could only be cleared or set by the RMU OPEN command ACCESS=[UN]RESTRICTED qualifier when it opened an Rdb database. If the restricted access bit is set in the database root, the RMU DUMP/HEADER command will show the following output when it displays the root parameters of an Rdb database. The ACCESS=[UN]RESTRICTED qualifier requires that the user has been granted RMU$ALTER privilege.

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

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

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

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

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

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

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

Bug 4374847

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

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

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

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

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

Usage Notes

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

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

  Views, global and local temporary tables, and information tables as well as misspelled names are ignored by this qualifier, even if they are named by the Validate qualifier. In such cases, no error is reported.
- If an invalid floating value is detected, then RMU will report the column name and the DBKEY of the affected row.
- The default Transaction_Type for the Validate action is PROTECTED READ mode. Use /TRANSACTION_TYPE=READ_ONLY to minimize locking.

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

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

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

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

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

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

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

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

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

```
$ rmu/show user/out=user.dat mf_personnel
$ search user.dat "active database users"
    - 5 active database users on this node
$ rmu/set database/abort mf_personnel
%RMU-I-MODIFIED, Database state modified
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
$ rmu/show user/out=user.dat mf_personnel
$ search user.dat "active database users"
%SEARCH-I-NOMATCHES, no strings matched
$ search forcex1.log abort
%RDMS-F-ABORTUSERS, database operator requested user termination
$ search forcex2.log abort
%RDMS-F-ABORTUSERS, database operator requested user termination
$ search forcex3.log abort
%RDMS-F-ABORTUSERS, database operator requested user termination
$ search forcex4.log abort
%RDMS-F-ABORTUSERS, database operator requested user termination
$ search forcex5.log abort
%RDMS-F-ABORTUSERS, database operator requested user termination
$
```

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

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

6.1.7 Smaller After Image Backup Files

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

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

7

Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.1

7.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.3.1

7.1.1 New and Modified Replication Network Error Handling Log Messages

Enhancement Bug 7282606

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

- Network retry count of # exceeded

  This message has been changed to:

  Network retry count of # exceeded; aborting...

  where # is the maximum number of times to try to re-establish the network connection. The attempt to re-establish the network connection has failed and been aborted after attempting to re-establish the connection the maximum number of times displayed in the message. The attempt to re-establish the network connection will be aborted before the retry count is reached if an attempt to shutdown and restart the network fails or an unrecoverable network error occurs during a retry attempt. The network retry count can be modified using the RDM$BIND_HOT_NETWORK_RETRY_COUNT logical.

- Unexpected loss of network connection; retrying...

  This new message has been added to indicate that the network connection has been lost and an attempt will be made to re-establish the network connection. The error status will have already been output in a previous log message.

- Unexpected loss of network connection, status: status_code - status_text; retrying...

  This new message has been added to indicate that an error has been returned which indicates that the network connection has been lost and an attempt will be made to re-establish the network connection. The error status code id is given followed by the error message name and text.

- Network connection failure, server "server_name"

  This message has been changed to:

  Network connection failure, server "server_name"; aborting...

  to indicate that the named Hot Standby server cannot connect to the network because of a network failure. The error status will have already been output in a previous log message.

- Unexpected network error status: status_code - status_text; aborting...
  Unexpected network error subcode: subcode_id - subcode_text

  This new message has been added to indicate that an unexpected and unrecoverable network error has been detected. The error status code id is given followed by the error message name and text. If there is a related secondary error status that status code is also given followed by the error message name and text.

- Network Connection lost, no attempt to reconnect will be made; aborting...

  This new message has been added to indicate that the connection to the network has been lost but in this case any retry attempt would fail so no attempt to re-establish the network connection will be made. The error status will have already been output in a previous log message.

- Network connection re-established

  This new message has been added to confirm that the network connection has been successfully re-established and the replication operation will continue.

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

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

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

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

Enhancement Bug 8945775

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

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

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

- INCLUDE="string"

  /REPORT_SCREEN=(INCLUDE="string")

  This keyword will include in the report only those screens with a screen name matching all or part of the specified "string". The string may contain wild card characters.

- EXCLUDE="string"

  /REPORT_SCREEN=(EXCLUDE="string")

  This keyword will exclude from the report only those screens with a screen name matching all or part of the specified "string". The string may contain wild card characters.

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

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

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

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

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

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

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

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

Enhancement Bug 1108555

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

The syntax of the OUTPUT qualifier is:

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

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

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

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

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

Directory DISK:[DIRECTORY]

TEST.LIS;1

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

TEST.LIS;1

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

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

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

Directory DISK:[DIRECTORY]

TEST.LIS;1

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

Directory DISK:[DIRECTORY]

TEST.LIS;1

Total of 1 file.
$ search test.lis RMU-I-COMPLETED
%RMU-I-COMPLETED, RESTORE operation completed at 27-JUN-2018 16:48:00.22
```

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

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

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

- Compression ZLIB level 6

  This first message shows the currently supported data compression algorithm "ZLIB" and the ZLIB level used, which will be an integer between 1 and 9. The higher the level number, the greater the compression but also the greater amount of CPU time spent doing the compression. The default level of 6 is a good trade off between the necessary CPU time and the amount of data compression.

- Data compressed by 39% (3956 KB in/2428 KB out)

  This second compression message, which immediately follows the first compression message, shows the amount of compression as a percent value based on the total number of input uncompressed bytes compared to the total number of output compressed bytes followed by the total number of uncompressed bytes and the total number of compressed bytes expressed in scaling units, which will vary depending on the amount of data compressed. The scaling units will be one of "Bytes", "KB" for kilobytes, "MB" for megabytes, "GB" for gigabytes or "TB" for Terabytes.

The following example shows the last portion of a log file named ABS_ 23CF6459.OUT created by an automatic backup server process with a process id of 23CF6459. The log messages show that this process has created the After Image Journal backup file AIJBCKCOMP.ABF;24 and that the data in the backup file was compressed using the ZLIB compression level 6 algorithm. The

compressed output data was 38% smaller than the uncompressed input data. The number of uncompressed input kilobytes is 3932 and the number of compressed output kilobytes in the AIJBCKCOMP.ABF;24 output AIJ backup file is 2445.

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

 4-OCT-2018 08:44:33.45 - AIJ Backup Server (ABS) activated
 4-OCT-2018 08:44:33.45 - Database is DEVICE:[DIRECTORY]AIJBCKCOMP.RDB;1
 4-OCT-2018 08:44:33.45 - Backing up AIJ 23
 4-OCT-2018 08:44:33.45 - By-sequence AIJ backup for sequence 23 to 23
%RDMS-I-OPERNOTIFY, system operator notification: AIJ backup operation started
 4-OCT-2018 08:44:33.45 - No Hot Standby servers active
%RDMS-I-AIJBCKSEQ, backing up after-image journal sequence number 23
%RDMS-I-LOGBCKAIJ, backing up after-image journal J4 at 08:44:33.45
%RDMS-I-LOGCREBCK, created backup file DEVICE:[DIRECTORY]AIJBCKCOMP.ABF;24
%RDMS-I-OPERNOTIFY, system operator notification: AIJ backup operation completed
 4-OCT-2018 08:44:34.34 - AIJ backup complete
 4-OCT-2018 08:44:34.34 - Compression ZLIB level 6
 4-OCT-2018 08:44:34.34 - Data compressed by 38% (3932 KB in/2445 KB out)
 4-OCT-2018 08:44:34.34 -  ELAPSED:    0 00:00:00.91  CPU: 0:00:00.79  BUFIO:
20  DIRIO: 356  FAULTS: 242
$
```

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

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

The syntax of the OUTPUT qualifier is:

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

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

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

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

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

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

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

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

Enhancement Bug 1108555

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

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

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

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

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

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

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

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

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

```
$ rmu/backup/parallel=executor=1/disk/execute/log -
    /output=parallel_log -
    /active_io=5 -
    /page_buffers=5 -
    /noquiet_point -
    /list_plan=mfp.plan -
    mf_personnel [.test1]mfp.rbf,[.test2],[.test3]
$
$ sear mfp.plan "output"
    Output = PARALLEL_LOG
$
$ rmu/backup/plan mfp.plan
$
$ type parallel_log.lis;2
WORKER_001: %RMU-I-BCKTXT_00, Backed up root file
DISK:[DIRECTORY]MF_PERSONNEL.RDB;1
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area (RDB$SYSTEM)
DISK:[DIRECTORY]MF_PERS_DEFAULT.RDA;1 at  3-AUG-2018 15:03:39.44
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area (RDB$SYSTEM)
 DISK:[DIRECTORY]MF_PERS_DEFAULT.RDA;1 at  3-AUG-2018 15:03:39.45
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area (EMPIDS_MID)
 DISK:[DIRECTORY]EMPIDS_MID.RDA;1 at  3-AUG-2018 15:03:39.45
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area
(MF_PERS_SEGSTR)
 DISK:[DIRECTORY]MF_PERS_SEGSTR.RDA;1 at  3-AUG-2018 15:03:39.45
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area (EMPIDS_MID)
 DISK:[DIRECTORY]EMPIDS_MID.RDA;1 at  3-AUG-2018 15:03:39.46
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area
(MF_PERS_SEGSTR)
 DISK:[DIRECTORY]MF_PERS_SEGSTR.RDA;1 at  3-AUG-2018 15:03:39.46
WORKER_001: %RMU-I-RESUME, resuming operation on volume 2 using DISK
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area
(SALARY_HISTORY)
 DISK:[DIRECTORY]SALARY_HISTORY.RDA;1 at  3-AUG-2018 15:03:39.47
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area (EMP_INFO)
 DISK:[DIRECTORY]EMP_INFO.RDA;1 at  3-AUG-2018 15:03:39.47
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area (EMPIDS_LOW)
 DISK:[DIRECTORY]EMPIDS_LOW.RDA;1 at  3-AUG-2018 15:03:39.47
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area
(SALARY_HISTORY)
 DISK:[DIRECTORY]SALARY_HISTORY.RDA;1 at  3-AUG-2018 15:03:39.48
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area (EMP_INFO)
 DISK:[DIRECTORY]EMP_INFO.RDA;1 at  3-AUG-2018 15:03:39.48
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area (EMPIDS_LOW)
 DISK:[DIRECTORY]EMPIDS_LOW.RDA;1 at  3-AUG-2018 15:03:39.48
WORKER_001: %RMU-I-RESUME, resuming operation on volume 3 using DISK
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area (JOBS)
 DISK:[DIRECTORY]JOBS.RDA;1 at  3-AUG-2018 15:03:39.49
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area (EMPIDS_OVER)
 DISK:[DIRECTORY]EMPIDS_OVER.RDA;1 at  3-AUG-2018 15:03:39.49
WORKER_001: %RMU-I-BCKTXT_02, Starting full backup of storage area (DEPARTMENTS)
 DISK:[DIRECTORY]DEPARTMENTS.RDA;1 at  3-AUG-2018 15:03:39.49
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area (JOBS)
 DISK:[DIRECTORY]JOBS.RDA;1 at  3-AUG-2018 15:03:39.50
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area
(EMPIDS_OVER)
 DISK:[DIRECTORY]EMPIDS_OVER.RDA;1 at  3-AUG-2018 15:03:39.50
WORKER_001: %RMU-I-BCKTXT_12, Completed full backup of storage area
(DEPARTMENTS)
 DISK:[DIRECTORY]DEPARTMENTS.RDA;1 at  3-AUG-2018 15:03:39.50
WORKER_001: %RMU-I-COMPLETED, BACKUP operation completed at
3-AUG-2018 15:03:39.51
%RMU-I-COMPLETED, BACKUP operation completed at  3-AUG-2018 15:03:39.51
$
$ directory parallel_log.lis;*
```

```
Directory DISK:[DIRECTORY]

PARALLEL_LOG.LIS;2  PARALLEL_LOG.LIS;1

Total of 2 files.
$
```

7.1.7 New CREATE OR REPLACE Support for PROFILE

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

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

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

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

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

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

7.1.8 New CREATE OR REPLACE Support for VIEW

Enhancement Bugs 3763080 and 4555512

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

Arguments

- OR REPLACE

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

Usage Notes

- If the view does not exist, then there must not be a table, sequence or synonym with the same name as this new view.
- If the view exists and the CREATE VIEW statement was used, then an error will be reported.
- If the view exists and the CREATE OR REPLACE VIEW statement was used and the name used is a synonym, then the view referenced by that synonym will be replaced.
- A view will be replaced if these conditions are met.

  1. There are no existing database object dependencies on the view.

     For example, there are no procedures, functions or other objects with references to the view and its columns.

     Note

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

  2. The existing dependencies are met by the new view definition.

     For example, if a view is referenced by a stored procedure then any column names referenced must exist after the replace is complete.

     Consider this example which attempts to reduce the columns of the view. The view CURRENT_INFO uses the view field SALARY_START, which is no longer present in the revised view definition.

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

3. There are no language semantic requirements due to the functionality used to reference the view.

   For example, an INSERT INTO statement that omits the column list or a SELECT * FROM statement have an implied column name list and also column ordering. This column name ordering must be maintained by the replace.

Note

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

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

Examples

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

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

7.1.9 New CREATE OR REPLACE Support for SEQUENCE

Enhancement Bug 3763080

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

Arguments

– OR REPLACE

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

Usage Notes

– You must have the CREATE database privilege to execute the CREATE SEQUENCE Statement. You must have the ALTER sequence privilege to execute the CREATE OR REPLACE SEQUENCE Statement on an existing sequence.

– If the sequence does not exist, then there must not be a table, synonym or view with the same name as this new sequence.

– If the sequence exists and the CREATE SEQUENCE Statement was used, then an error will be reported.

– If the sequence exists and the CREATE OR REPLACE SEQUENCE Statement was used and the name used is a synonym, then the sequence referenced by that synonym will be replaced.

– A system sequence (created internally by Oracle Rdb), or column identity sequence (created by the IDENTITY or GENERATED ... AS IDENTITY clause), may not be replaced.

– If a sequence is replaced, the START WITH value will be reset to the value specified by the CREATE SEQUENCE Statement or the default based on the other clauses in the statement.

- Exclusive access is required if a sequence is replaced. No active queries or other users may have access to that sequence.
- The access control list (ACL) on the sequence is propagated (saved and restored) from the old sequence by the REPLACE action.
- The audit and alarm settings for the sequence are propagated (saved and restored) from the old sequence by the REPLACE action.
- Any comment on the sequence is propagated (saved and restored) from the old sequence by the REPLACE action unless there is a COMMENT IS clause specified by the CREATE OR REPLACE Statement. Comments can be created by the COMMENT IS clause of the CREATE or ALTER SEQUENCE Statement or by the COMMENT ON SEQUENCE Statement.

Examples

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

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

-- show that comment and ACL are propagated by OR REPLACE
show sequence DEPT_ID;
     DEPT_ID
 Sequence Id: 3
 Initial Value: 10
 Minimum Value: 1
 Maximum Value: 9223372036854775806
 Next Sequence Value: 10
 Increment by: 1
 Cache Size: 20
 No Order
 Cycle
 No Randomize
 Comment:       revised; new departments get a unique number

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

7.1.10 New Options for SET LOGFILE Statement

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

- LOGFILE quoted-filespec

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

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

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

- CACHE

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

- NOCACHE

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

- ECHO

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

- NOECHO

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

- LARGE_FILE

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

- SHARED

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

7.1.11 New UNDECLARE CURSOR Statement

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

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

Environment

You can use the UNDECLARE CURSOR statement:

- In interactive SQL

Format

UNDECLARE CURSOR <cursor-name>

Arguments

cursor-name

Specifies the name of the declared cursor.

Example

Example of using the Undeclare Cursor statement.

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

7.1.12 Enhanced LIKE Table Support in CREATE TABLE Statement

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

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

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

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

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

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

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

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

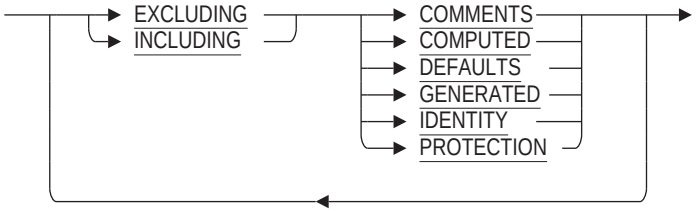
By default, GENERATED, AUTOMATIC, IDENTITY and COMPUTED columns are not copied but columns representing the same data types are created instead.

Syntax

column-constraint-list =

ansi-like-table-clause =

like-attributes =



Usage Notes

- When using the LIKE clause to copy a table definition, the creator of the new table must have REFERENCES or SELECT privilege granted for the referenced table.
- The LIKE clause can be used multiple times within a CREATE TABLE statement. However, if the copied tables include any duplicate column names, then an error will be reported. Only one IDENTITY column can be defined or inherited. Use the INCLUDING IDENTITY clause, if necessary, to inherit the attributes from the referenced table.

  The default behavior is EXCLUDING COMPUTED, GENERATED, IDENTITY column details. In this case, non-generated columns will be created which contain the same data type attributes. Default values defined for the source tables are not automatically inherited. Use the INCLUDING DEFAULTS clause to control this behavior.

  Note: For backward compatibility with previous versions of Oracle Rdb, the LIKE clause used outside the column-constraint-list defaults to INCLUDING GENERATED, INCLUDING IDENTITY, INCLUDING COMPUTED and INCLUDING DEFAULTS. The like-attributes may not be specified in this location and therefore these defaults may not be changed.
- The clauses EXCLUDING GENERATED or INCLUDING GENERATED apply to columns defined using the GENERATED ... AS (expr) and AUTOMATIC ... AS (expr) syntax. When EXCLUDING is used or implied, the generated (or automatic) column is converted to a simple base column with the same data types.
- The clauses EXCLUDING IDENTITY or INCLUDING IDENTITY apply to columns defined using the GENERATED ... AS IDENTITY and IDENTITY (...) syntax. When EXCLUDING is used or implied, the identity column is converted to a simple base column with the same data types.

- The clauses EXCLUDING COMPUTED or INCLUDING COMPUTED apply to columns defined using the COMPUTED BY expr syntax. When EXCLUDING is used or implied, the computed by column is converted to a simple base column with the same data types. Note that the column will require space in the defined table, which isn't true for COMPUTED BY columns.
- When the LIKE clause is used within the column-constraint-list, then EXCLUDING DEFAULTS is assumed. Use the INCLUDING DEFAULTS if you wish the inherited columns to have DEFAULTS inherited from the source table.
- The LIKE clause is only used to inherit the column definitions from the referenced table. Once the table is created with LIKE clauses, subsequent changes to the source table are not propagated to the created tables.

Examples

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

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

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

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

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

SQL>
```

7.1.13 New TO_DSINTERVAL and TO_YMINTERVAL Functions

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

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

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

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

Syntax

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


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

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

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

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

Duration Format

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

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

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

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

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

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

Some examples:

- "P3Y6M4DT12H30M5.6S" represents a duration of three years, six months, four days, twelve hours, thirty minutes, and five point 6 seconds.
- "P1Y" represents a duration of one year.
- "PT50000S" represents a duration of fifty thousand seconds, which will be represented as thirteen hours fifty three minutes and twenty seconds.

- "P1MT1M" represents one month and one minute. The "T" separator is required to avoid ambiguity between months and minutes. This duration is accepted by Oracle Rdb as a year-month interval as input TO_YMINTERVAL but the low precision fields (D, H, M, S) will be discarded.

7.1.14 New CREATE DEFAULT AUDIT Statement

CREATE DEFAULT AUDIT Statement

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

This statement can be used to create templates for:

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

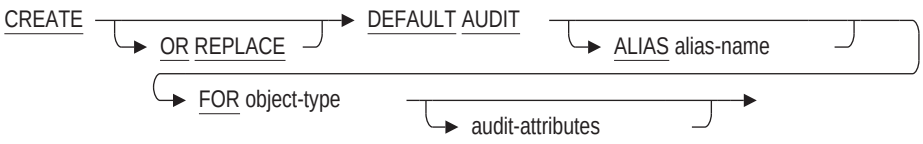
See also ALTER DEFAULT AUDIT Statement and the DROP DEFAULT AUDIT Statement.

Environment

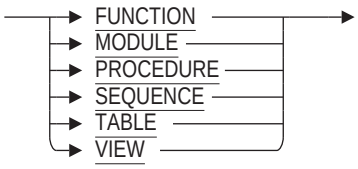
You can use the CREATE DEFAULT AUDIT statement:

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

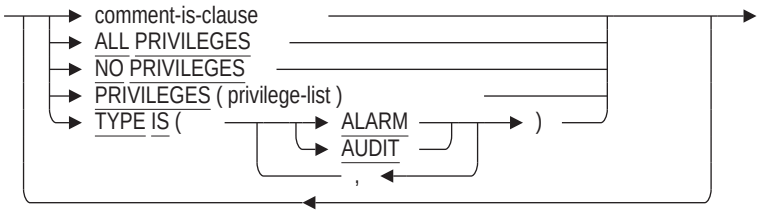
Syntax



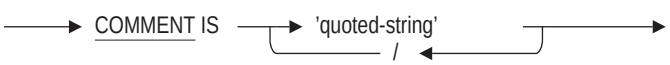
object-type =



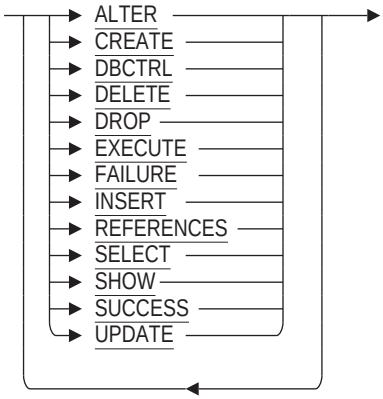
audit-attributes =



comment-is-clause =



privilege-list =



Arguments

- ALIAS aliasname

  The name of the database alias if there is no default database for this session.

- COMMENT IS

  Adds a comment to the template object.

- FOR object-type

  The type of template.

- PRIVILEGES (privilege-list)
  ALL PRIVILEGES
  NO PRIVILEGES

  Specifies the privileges that trigger audit or alarm actions.

  The ALL PRIVILEGES clause is equivalent to specifying PRIVILEGES (ALL). NO PRIVILEGES is the default if none of these clauses is specified.

- TYPE IS (ALARM)
  TYPE IS (AUDIT)
  TYPE IS (ALARM, AUDIT)

Defines the scope for the privileges checking. The default if this clause is not specified is both ALARM and AUDIT.

Usage Notes

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

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

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

  This example grants privileges to a role (rights identifier) and a user:

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

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

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

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

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

7.1.15 New ALTER DEFAULT AUDIT Statement

ALTER DEFAULT AUDIT Statement

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

This statement can be used to alter templates for:

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

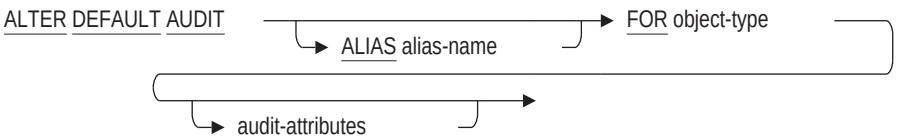
See also CREATE DEFAULT AUDIT Statement and the DROP DEFAULT AUDIT Statement.

Environment

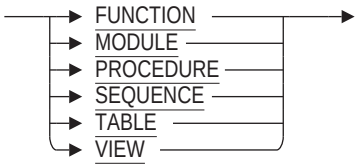
You can use the ALTER DEFAULT AUDIT statement:

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

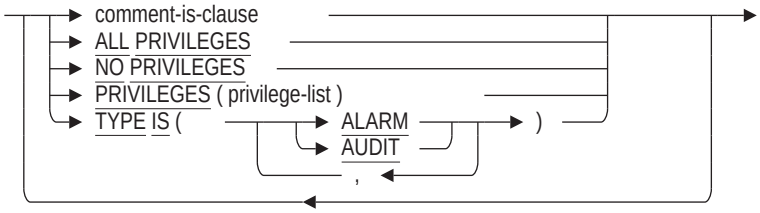
Syntax



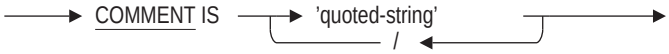
object-type =



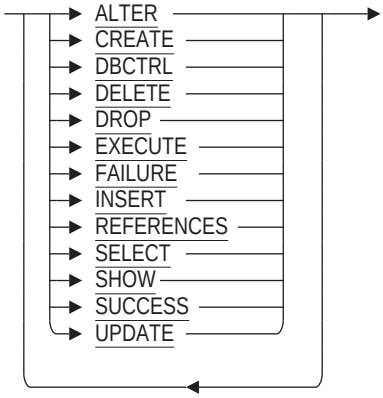
audit-attributes =



comment-is-clause =



privilege-list =



Arguments

- ALIAS aliasname

  The name of the database alias if there is no default database for this session.

- COMMENT IS

  Adds a comment to the template object.

- FOR object-type

  The type of template.

- PRIVILEGES (privilege-list)
  ALL PRIVILEGES
  NO PRIVILEGES

  Specifies the privileges that trigger audit or alarm actions.

The ALL PRIVILEGES clause is equivalent to specifying PRIVILEGES (ALL). NO PRIVILEGES is the default if none of these clauses is specified.

- TYPE IS (ALARM)
  TYPE IS (AUDIT)
  TYPE IS (ALARM, AUDIT)

  Defines the scope for the privileges checking. The default if this clause is not specified is both ALARM and AUDIT.

Usage Notes

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

7.1.16 New DROP DEFAULT AUDIT Statement

DROP DEFAULT AUDIT Statement

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

This statement can be used to drop templates for:

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

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

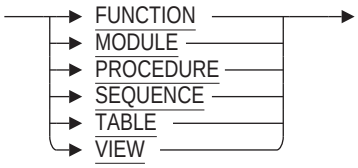
Environment

You can use the DROP DEFAULT AUDIT statement:

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

Syntax

object-type =



Arguments

- ALIAS aliasname

  The name of the database alias if there is no default database for this session.

Usage Notes

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

7.1.17 New SESSION and GLOBAL Attributes for Sequences

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

- SESSION

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

  You may not specify both SESSION and GLOBAL.

- GLOBAL

  Specify GLOBAL to create a global, or regular, sequence. This is the default.

  You may not specify both GLOBAL and SESSION.

The following example shows the creation of a SESSION sequence.

```
SQL> create sequence Example
cont>     nomaxvalue
cont>     session
cont>     start with 456;
SQL>
SQL> show sequence Example;
     EXAMPLE
 Sequence Id: 4
 Initial Value: 456
 Minimum Value: 1
 Maximum Value: (none)
 Next Sequence Value: 456
 Increment by: 1
 Cache Size: 20
 No Order
 No Cycle
 No Randomize
 Session (local temporary sequence)
 Wait
```

```
SQL>
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 269 committed
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;5" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 5 data records written,
 specified reopen count is 5
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;6" created
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 270 committed
%RMU-I-LOGRECSTAT, transaction with TSN 271 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 272 committed
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;6" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 5 data records written,
 specified reopen count is 5
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 260 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;7" created
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 274 committed
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 275 committed
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;7" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 5 data records written,
 specified reopen count is 5
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;8" created
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 276 committed
%RMU-I-LOGRECSTAT, transaction with TSN 265 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 278 committed
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;8" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 5 data records written,
 specified reopen count is 5
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]INVOICES.DAT;9" created
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 279 committed
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 273 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 281 committed
%RMU-I-LOGRECSTAT, transaction with TSN 283 committed
%RMU-I-LOGRECSTAT, transaction with TSN 282 committed
%RMU-I-LOGRECSTAT, transaction with TSN 286 committed
%RMU-I-LOGRECSTAT, transaction with TSN 288 committed
%RMU-I-EXTSRTSTAT, Records:7 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 289 committed
%RMU-I-LOGRECSTAT, transaction with TSN 291 committed
%RMU-I-AIJMODSEQ, next AIJ file sequence number will be 1
%RMU-I-LOGSUMMARY, total 30 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
------------------------------------------------------------------------------
ELAPSED:    0 00:00:00.09  CPU: 0:00:00.01  BUFIO: 167  DIRIO: 105  FAULTS: 186
Table "INVOICES" : 35 records written (33 modify, 2 delete)
Table "COSTS" : 9 records written (9 modify, 0 delete)
Total : 44 records written (42 modify, 2 delete)
$
```

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

```
$ rmu/unload/after_journal/statistics/FORMAT=DUMP/TRACE/LOG -
        accounting.rdb accounting_jrnl.aij -
        /table=(name=invoices, -
        output=reports.dat, -
        REOPEN=(RECORDS=5), -
        table_definition=invoices_def, -
        record_definition=rdb_lm_invoices ) -
        /table=(name=costs, -
        record_definition=rdb_lm_costs, -
        table_definition=costs_def, -
        output=reports.dat)
%RMU-I-UNLAIJFL, Unloading table INVOICES to DEVICE:[DIRECTORY]REPORTS.DAT;1
%RMU-I-UNLAIJFL, Unloading table COSTS to DEVICE:[DIRECTORY]REPORTS.DAT;1
%RMU-I-LOGOPNAIJ, opened journal file
 DEVICE:[DIRECTORY]ACCOUNTING_JRNL.AIJ;1 at 29-JAN-2019 13:45:07.66
%RMU-I-AIJRSTSEQ, journal sequence number is "0"
29-JAN-2019 13:45:07.66 Starting at offline open record sequence number 0
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 256 committed
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 257 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;1" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 6 data records written,
 specified reopen count is 5
%RMU-I-LOGRECSTAT, transaction with TSN 258 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;2" created
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 259 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;2" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 6 data records written,
 specified reopen count is 5
%RMU-I-LOGRECSTAT, transaction with TSN 261 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;3" created
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 262 committed
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 263 committed
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
```

```
%RMU-I-LOGRECSTAT, transaction with TSN 264 committed
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;3" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 5 data records written,
 specified reopen count is 5
%RMU-I-LOGRECSTAT, transaction with TSN 266 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;4" created
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 267 committed
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 268 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;4" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 7 data records written,
 specified reopen count is 5
%RMU-I-LOGRECSTAT, transaction with TSN 269 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;5" created
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 270 committed
%RMU-I-LOGRECSTAT, transaction with TSN 271 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 272 committed
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LMREOPENCLOSE, Reopen record count reached,
 output file "DEVICE:[DIRECTORY]REPORTS.DAT;5" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 6 data records written,
 specified reopen count is 5
%RMU-I-LOGRECSTAT, transaction with TSN 260 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;6" created
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 274 committed
%RMU-I-EXTSRTSTAT, Records:1 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 275 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LMREOPENCLOSE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;6" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 6 data records written,
 specified reopen count is 5
%RMU-I-LOGRECSTAT, transaction with TSN 276 committed
%RMU-I-LOGRECSTAT, transaction with TSN 265 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached,
 output file "DEVICE:[DIRECTORY]REPORTS.DAT;7" created
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 278 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LMREOPENCLOSE, Reopen record count reached,
 output file "DEVICE:[DIRECTORY]REPORTS.DAT;7" closed
%RMU-I-LMREOPENCOUNTS, Table "INVOICES" : 5 data records written,
 specified reopen count is 5
%RMU-I-LOGRECSTAT, transaction with TSN 279 committed
%RMU-I-LMREOPENCREATE, Reopen record count reached, output file
 "DEVICE:[DIRECTORY]REPORTS.DAT;8" created
%RMU-I-EXTSRTSTAT, Records:2 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 273 committed
%RMU-I-EXTSRTSTAT, Records:3 Merges:0 Nodes:0 WorkAlq:0
%RMU-I-LOGRECSTAT, transaction with TSN 281 committed
%RMU-I-LOGRECSTAT, transaction with TSN 283 committed
%RMU-I-LOGRECSTAT, transaction with TSN 282 committed
%RMU-I-LOGRECSTAT, transaction with TSN 286 committed
%RMU-I-LOGRECSTAT, transaction with TSN 288 committed
```

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

8

Documentation Corrections, Additions and Changes

This chapter provides corrections for documentation errors and omissions.

8.1 Documentation Corrections

8.1.1 Oracle Rdb Release 7.3.x.x New Features Document Added

A new document has been created which contains all of the New Features Chapters from all Rdb 7.3 Release Notes. This document will be included in saveset A of the Rdb kit. It is called RDB_NEWFEATURES_73xx and will be available in postscript, text and PDF format. This will provide customers with one document to reference to find out about all new features that have been added to the Rdb 7.3 releases.

8.1.2 Undocumented /TRANSACTION=EXCLUSIVE Option for RMU Populate_Cache

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

- Transaction_Type=option

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

  - Automatic
  - Exclusive
  - Read_Only
  - Noread_Only

  You must specify an option if you use this qualifier.

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

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

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

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

Note

Using Transaction_Type=Exclusive might interfere with other actions occurring on the database; especially note that ONLINE backup and some actions of the Row Cache server are incompatible with this Transaction_ Type.

8.1.3 Action of DECLARE TRANSACTION When SQL$DATABASE is Defined

Bug 5705204

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

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

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

8.1.4 RDB$USAGE Field Values

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

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

Many reported errors include text from the RDB$USAGE field to explain the type of dependency preventing the DROP from succeeding. These text strings are described in Table 8–1, Rdb$USAGE Field Values.

Table 8–1 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

(continued on next page)

Table 8–1 (Cont.) Rdb$USAGE Field Values

Field Value	Description
Storage Map	Storage map references table and column
View	View definition requires table, view, column, sequence or function
View Field	View column requires table, view, column, sequence or function
Trigger	Trigger definition requires table, view, column, sequence or function
RelConstraint	A table (relation) constraint references a table
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

(continued on next page)

Table 8–1 (Cont.) Rdb$USAGE Field Values

Field Value	Description
Storage Map Function	When CREATE STORAGE MAP is executed, a system routine is created to reflect the mapping. Those column dependencies are recorded

8.1.5 Updated Documentation for RDMS$BIND_CODE_OPTIMIZATION Logical

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

- Alpha Systems

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

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

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

  For more information on 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...
```

8.1.6 New RDM$BIND_AIJ_BUFFER_POOL_COUNT Logical

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

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

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

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

```
$ 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)
$!
```

8.1.7 RDMSTT Image Optionally Installed

Bug 3981903

If you plan on using the cluster capability of RMU/SHOW STATISTICS, Oracle recommends that you install the RDMSTT73.EXE image on OpenVMS with the appropriate privileges.

The RMONSTART73.COM command procedure provided by the Oracle Rdb installation allows the system manager to optionally install the RDMSTT73.EXE image at monitor startup time. To take advantage of this, you will need to edit the SYS$STARTUP:RMONSTART73.COM procedure and remove the comment characters from the following lines:

```
$ ! DEFX SYS$COMMON:[SYSEXE]RDMSTT73.EXE
$ !     REMOVEX
$ !     ADDX /OPEN/HEAD/PROT/PRIV=(CMKRNL,SYSPRV,SHARE)
```

Also, edit the command procedure SYS$MANAGER:RMONSTOP73.COM and remove the comment characters from the following lines:

```
$ ! DEFX SYS$COMMON:[SYSEXE]RDMSTT73.EXE
$ !     REMOVEX
```

8.1.8 Clarification on the Affect of the RMU/OPEN/ACCESS=RESTRICTED Command

The command RMU/OPEN/ACCESS=RESTRICTED can be used to prevent non-privileged users attaching to the database while database maintenance is being performed.

This command sets a permanent attribute in the database root that must be cleared by a subsequent RMU/OPEN/ACCESS=UNRESTRICTED command. This is true even if the database is defined OPEN IS AUTOMATIC. An RMU/CLOSE might be required before the RMU/OPEN can be executed.

The following example shows the reported error when this attribute is active; NOPRIV with an explanation DBNOTOPEN.

```
SQL> att 'f personnel user ''dbuser2'' using ''*******'' ';
%SQL-F-ERRATTDEC, Error attaching to database personnel
-RDB-E-NO_PRIV, privilege denied by database facility
-RDMS-F-DBNOTOPEN, database is not open for access
-RDB-F-ON_DB, on database _$1$DGA174:[TESTING]PERSONNEL.RDB;1
SQL>
```

The database administrator must be granted RMU$OPEN on the database root file, as well as having DBADM on the database (this might be inherited from OpenVMS override privileges) to execute this command.

The RMU/DUMP/HEADER command will show this setting as seen in the following output:

```
$ rmu/dump/header=ROOT_RECORD PERSONNEL
*-------------------------------------------------------------------------------
* Oracle Rdb V7.3-210                                    22-NOV-2016 14:27:23.59
*
* Dump of Database header
*     Database: USER2:[TESTING]PERSONNEL.RDB;1
*
*-------------------------------------------------------------------------------

Database Parameters:
    Root filename is "USER2:[TESTING]PERSONNEL.RDB;1"

Oracle Rdb specific root record

    Dbkey for Oracle Rdb bootstrap page is 8:462:0
    Access restricted to privileged users
    Current metadata version is 26
    Database has never been backed up
    Database has never been incrementally restored
    Database has never been fully restored
    Database has never been verified
    Database has never been altered
```

8.1.9 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 8–2 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 8–2 Optional System Tables and Their Storage Map Names

Table Name	Storage Map Name	Associated Feature
RDB$CATALOG_ SCHEMA	RDB$CATALOG_ SCHEMA_MAP	Multischema databases

(continued on next page)

Table 8–2 (Cont.) Optional System Tables and Their Storage Map Names

Table Name	Storage Map Name	Associated Feature
RDB$CHANGES	RDB$CHANGES_MAP	Replication Option for Rdb
RDB$CHANGES_MAX_ TSER	RDB$CHANGES_MAX_ TSER_MAP	Replication Option for Rdb
RDB$SYNONYMS	RDB$SYNONYMS_MAP	Multischema databases
RDB$TRANSFERS	RDB$TRANSFERS_MAP	Replication Option for Rdb
RDB$TRANSFER_ RELATIONS	RDB$TRANSFER_ RELATIONS_MAP	Replication Option for Rdb
RDB$WORKLOAD	RDB$WORKLOAD_MAP	Workload Collection was 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.

8.1.10 Logical Name RDM$BIND_HOT_NETWORK_OBJECT Was Not Documented

Bug 2534163

The following logical name was not previously documented for Oracle Rdb.

- RDM$BIND_HOT_NETWORK_OBJECT

  The name of the network object for the AIJSERVER process on the remote standby database. This logical can be used with the following transports: DECnetIV, DECnetOSI, and HPE TCP/IP Services for OpenVMS.

  Specify RDMAIJ<version-number> for Oracle Rdb databases (example: RDMAIJ73).

Please note that the RMU/SHOW LOGICAL/UNDEFINED/DESCRIPTION command will display the description for this logical name.

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

Bug 12595718

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

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

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

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

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

```
$ sql$
SQL> attach 'file mf_personnel';
SQL> select RDB$SEQUENCE_NAME, RDB$NEXT_SEQUENCE_VALUE
cont> from rdb$sequences;
 RDB$SEQUENCE_NAME                RDB$NEXT_SEQUENCE_VALUE
 S                                                     21
1 row selected
exit;
$ delete mf_personnel.rdb;*
$ rmu/restore/only_root/log/NOSET_TSN mf_personnel
%RMU-I-AIJRSTBEG, restoring after-image journal "state" information
%RMU-I-AIJRSTJRN, restoring journal "AIJ1" information
%RMU-I-AIJRSTSEQ, journal sequence number is "0"
%RMU-I-AIJRSTSUC, journal "AIJ1" successfully restored from file
 "DEVICE:[DIRECTORY]AIJ_ONE.AIJ;1"
%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;
$
```

8.1.12 Oracle Rdb Position on NFS Devices

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

Active System Files

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

Archived Data Files

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

Other Miscellaneous Files

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

Setting Up NFS

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

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

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

Read Performance Issues

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

Disclaimer

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

8.1.13 RDM$BIND_STAREA_EMERGENCY_DIR Logical Name

Bugs 19545970 and 3682207

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

RDM$BIND_STAREA_EMERGENCY_DIR provides an alternate device and/or directory specification for the standby that can replace all or part of the master's file specification. Without the logical, the device and directory of the new storage area issued from the master must exist and match exactly on the standby. For example, on the master database we want to create a new starea, 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.

8.1.14 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.
```

8.1.15 Undocumented Hot Standby Logical Names

Bug 3264793

Table 8–3 Hot Standby Logical Names

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

(continued on next page)

Table 8–3 (Cont.) Hot Standby Logical Names

Logical Name Description	Default Value	Minimum Value	Maximum Value
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
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.			

8.1.16 Missing Documentation for the TRANSACTION_TYPE Keyword for GET DIAGNOSTICS

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

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

The result data type is CHAR (31).

Examples

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

Example 8–1 Example: Using TRANSACTION_TYPE to control actions of a procedure

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

8.1.17 Clarification on Using the RMU/UNLOAD TRIM=TRAILING Option

The following example shows that unexpected results may occur with the RMU/UNLOAD command Trim option when spaces are unloaded from an Oracle Rdb database.

Create a table in an Rdb database with two character columns and insert spaces and other character data into the table fields.

```
SQL> create database filename testdb;
create table tab1(col1 char(2), col2 char(2));
insert into tab1 values ('  ', '  ');
insert into tab1 values ('AB', '  ');
insert into tab1 values ('  ', 'CD');
insert into tab1 values ('A ', 'C ');
commit;
```

Unload the character field data from the table, specifying the Trim=trailing option to eliminate trailing spaces but do not specify prefix or suffix delimiter values.

```
 $ rmu/unload/record=(file=tab1, -
   format=delimited_text, prefix="", suffix="", separator="|", -
   null="NULL", trim=trailing) testdb tab1 tab1
```

The trailing spaces are eliminated from the unload file since the Trim=trailing option was used.

```
$ ty tab1.unl
|
AB|
|CD
A|C
```

Now load the unloaded data back into the database table from the unload file.

```
$ rmu/load/log/record=(file=tab1, -
   format=delimited_text, prefix="", suffix="", separator="|", -
   null="NULL") testdb tab1 tab1
```

This is the result:

```
SQL> att 'f testdb';
SQL> select '>' || col1 || '<   ','>' || col2 || '<' from tab1;

 >  <      >  <
 >AB<      >  <
 >  <      >CD<
 >A <      >C <
 >  <      NULL
 >AB<      NULL
 >  <      >CD<
 >A <      >C <
8 rows selected
```

The first and second row, which originally contained two spaces in COL2, are now set to NULL. This happens because of the use of the option Trim=trailing in the RMU/UNLOAD command.

Because neither prefix nor suffix characters are specified in the RMU/UNLOAD command, it cannot be determined whether values existed in the character fields which only contained spaces or if these fields were flagged as NULL fields in the database.

The trailing column is set to NULL as described by the Oracle Rdb RMU Reference Manual which states:

"If the final column or columns of a record are to be set to NULL, you only have to specify data for the column up to the last non-null column. See the Examples section for an example of each of these methods of storing the NULL value."

Therefore, a trailing empty field will be null. Inner columns will be null if set to the string specified by the NULL option.

To avoid this result, you could eliminate the TRIM option in the RMU/UNLOAD command, or if you need the TRIM option then you can avoid this result by specifying a character value for the PREFIX and SUFFIX separator options for both the RMU/LOAD and the RMU/UNLOAD commands as in the following example.

```
$ rmu/unload/record=(file=tab1, -
   format=delimited_text, prefix="*", suffix="*", separator="|", -
   null="NULL", trim=trailing) testdb tab1 tab1
$
$ ty tab1.unl
**|**
*AB*|**
**|*CD*
*A*|*C*
$
$ rmu/load/log/record=(file=tab1, -
   format=delimited_text, prefix="*", suffix="*", separator="|", -
   null="NULL") testdb tab1 tab1
```

8.1.18 Corrections to the EDIT STRING Documentation

Bugs 17365476 and 17365597

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

  This is the corrected text:

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

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

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

  rather than

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

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

8.1.19 Changes and Improvements to the Rdb Optimizer and Query Compiler

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

- CAST function elimination

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

  This replacement includes the following:

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

- CAST of string literals to DATE (ANSI), DATE (VMS), TIME, TIMESTAMP and INTERVAL can be processed at compile time. For example, CAST('2013-1-1' AS DATE ANSI) is implicitly converted to a DATE literal DATE'2013-1-1'.
- CAST of small integer values is now done by the compiler. For example, CAST(1 AS SMALLINT) can be performed at compile time.
- CAST of fixed length (CHAR) literal strings to varying length strings (VARCHAR) is now processed by the compiler if the character set is the same and the target VARCHAR is long enough to hold the source string, as seen in the following example:

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

- Constant Folding

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

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

  This optimization includes the following:

  * Addition (+)
  * Subtraction (-)
  * Multiplication (*)
  * Division (/)

    Note that division is not performed at compile time if the divisor is a literal zero (0). Operations which are coded to explicitly divide by zero are probably expected to produce an error at runtime. Although using the SQL SIGNAL statement is now preferred, this technique has been used to terminate procedures when an incorrect input is encountered.

- Algebraic Rules

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

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

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

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

- Simple Predicate Elimination

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

- Not Nullable Aware

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

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

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

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

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

- Replace comparisons with NULL

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

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

- Predicate Pruning

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

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

- CASE Expression Pruning

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

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

- IN Operator Simplification

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

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

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

The following examples show the types of likely results.

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

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

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

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

REWRITE Flag

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

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

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

- BOOLEANS
- IN_CLAUSE
- IS_NULL
- UNKNOWN

8.1.20 Required Privileges for AUTHORIZATION Clause of CREATE MODULE

The following usage note is missing from the SQL Reference Manual, under the CREATE MODULE Statement.

- When the AUTHORIZATION clause is used, the definer of the module is granting his/her own privileges to the specified username so that tables, columns, sequences, procedures and functions are accessed as though accessed by the definer.

  The AUTHORIZATION is expected to be the session user, or an OpenVMS rights identifier granted to that user (when SECURITY CHECKING IS EXTERNAL). If the session is run with one of the following OpenVMS privileges, then any user or rights identifier can be referenced: SYSPRV, BYPASS or IMPERSONATE.

  Note

  The OpenVMS IMPERSONATE privilege can be used to override the checking for Oracle Rdb Release 7.2.5.1 and later versions.

8.1.21 Sorting Capabilities in Oracle Rdb

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

QSORT

Use of QSORT preempts use of all other sorting algorithms. The QSORT algorithm is used if sorting is being done on a single key and if only a small amount of data is involved. The reason for this is that the other sorting algorithms, while using more efficient methods, have a certain amount of overhead associated with setting them up and with being general purpose routines.

QSORT is used by default if:

- There is a single sort key.
- The number of rows to be sorted is 5000 or fewer.
- The sort key is not floating point (REAL, FLOAT, or DOUBLE PRECISION).

How to Alter QSORT Usage

To change the usage of QSORT to evaluate behavior with other parameters, define a new row limit as follows:

```
$ DEFINE RDMS$BIND_MAX_QSORT_COUNT m
```

The default value is 5000 rows.

Note

Defining the logical RDMS$BIND_MAX_QSORT_COUNT as 63 will return QSORT behavior to that used by prior releases of Oracle Rdb V7.2.

To disable QSORT because of either anomalous or undesirable performance, the user can define the following logical to the value zero, in which case the VMS SORT interface is always used.

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

8.1.22 RMU /VERIFY Process Quotas and Limits Clarification

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

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

8.1.23 RDM$BIND_MAX_DBR_COUNT Documentation Clarification

Bugs 1495227 and 3916606

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

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

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

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

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

Per-Database Value

The RDM$BIND_MAX_DBR_COUNT logical name specifies the maximum number of database recovery processes to run at once for each database. For example, if there are 10 databases being recovered

and the value for the RDM$BIND_MAX_DBR_COUNT logical name is 8, up to 80 database recovery processes would be started by the monitor after a node failure.

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

8.1.24 Database Server Process Priority Clarification

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

Individual server priorities can be explicitly controlled via system-wide logical names as described in Table 8–4.

Table 8–4 Server Process Priority Logical Names

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

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

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

8.1.25 RDM$BIND_LOCK_TIMEOUT_INTERVAL Overrides the Database Parameter

Bug 2203700

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

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

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

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

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

8.1.26 Missing Tables Descriptions for the RDBEXPERT Collection Class

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

Table 8–5 shows the TRANS_TPB table.

Table 8–5 Columns for Table EPC$1_221_TRANS_TPB

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

Table 8–6 shows the TRANS_TPB_ST table. An index is provided for this table. It is defined with column STR_ID, duplicates are allowed, and the type is sorted.

Table 8–6 Columns for Table EPC$1_221_TRANS_TPB_ST

Column Name	Data Type	Domain
STR_ID	INTEGER	STR_ID_DOMAIN
SEGMENT_NUMBER	SMALLINT	SEGMENT_NUMBER_DOMAIN
STR_SEGMENT	VARCHAR(128)	

8.1.27 Missing Columns Descriptions for Tables in the Formatted Database

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

Table 8–7 shows the DATABASE table.

Table 8–7 Columns for Table EPC$1_221_DATABASE

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

Table 8–8 shows the REQUEST_ACTUAL table.

Table 8–8 Columns for Table EPC$1_221_REQUEST_ACTUAL

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_ DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_START	DATE VMS	
TIMESTAMP_END	DATE VMS	
DBS_READS_START	INTEGER	
DBS_WRITES_START	INTEGER	
RUJ_READS_START	INTEGER	
RUJ_WRITES_START	INTEGER	
AIJ_WRITES_START	INTEGER	
ROOT_READS_START	INTEGER	
ROOT_WRITES_START	INTEGER	
BUFFER_READS_START	INTEGER	
GET_VM_BYTES_START	INTEGER	
FREE_VM_BYTES_START	INTEGER	
LOCK_REQS_START	INTEGER	

(continued on next page)

Table 8–8 (Cont.) Columns for Table EPC$1_221_REQUEST_ACTUAL

Column Name	Data Type	Domain
REQ_NOT_QUEUED_START	INTEGER	
REQ_STALLS_START	INTEGER	
REQ_DEADLOCKS_START	INTEGER	
PROM_DEADLOCKS_START	INTEGER	
LOCK_RELS_START	INTEGER	
LOCK_STALL_TIME_START	INTEGER	
D_FETCH_RET_START	INTEGER	
D_FETCH_UPD_START	INTEGER	
D_LB_ALLOK_START	INTEGER	
D_LB_GBNEEDLOCK_START	INTEGER	
D_LB_NEEDLOCK_START	INTEGER	
D_LB_OLDVER_START	INTEGER	
D_GB_NEEDLOCK_START	INTEGER	
D_GB_OLDVER_START	INTEGER	
D_NOTFOUND_IO_START	INTEGER	
D_NOTFOUND_SYN_START	INTEGER	
S_FETCH_RET_START	INTEGER	
S_FETCH_UPD_START	INTEGER	
S_LB_ALLOK_START	INTEGER	
S_LB_GBNEEDLOCK_START	INTEGER	
S_LB_NEEDLOCK_START	INTEGER	
S_LB_OLDVER_START	INTEGER	
S_GB_NEEDLOCK_START	INTEGER	
S_GB_OLDVER_START	INTEGER	
S_NOTFOUND_IO_START	INTEGER	
S_NOTFOUND_SYN_START	INTEGER	
D_ASYNC_FETCH_START	INTEGER	
S_ASYNC_FETCH_START	INTEGER	
D_ASYNC_READIO_START	INTEGER	
S_ASYNC_READIO_START	INTEGER	
AS_READ_STALL_START	INTEGER	
AS_BATCH_WRITE_START	INTEGER	
AS_WRITE_STALL_START	INTEGER	
BIO_START	INTEGER	
DIO_START	INTEGER	
PAGEFAULTS_START	INTEGER	
PAGEFAULT_IO_START	INTEGER	
CPU_START	INTEGER	

(continued on next page)

Table 8–8 (Cont.) Columns for Table EPC$1_221_REQUEST_ACTUAL

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

(continued on next page)

Table 8–8 (Cont.) Columns for Table EPC$1_221_REQUEST_ACTUAL

Column Name	Data Type	Domain
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 8–9 shows the TRANSACTION table.

Table 8–9 Columns for Table EPC$1_221_TRANSACTION

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_ DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_START	DATE VMS	

(continued on next page)

Table 8–9 (Cont.) Columns for Table EPC$1_221_TRANSACTION

Column Name	Data Type	Domain
TIMESTAMP_END	DATE VMS	
CLIENT_PC_START	INTEGER	
STREAM_ID_START	INTEGER	
LOCK_MODE_START	INTEGER	
TRANS_ID_START	VARCHAR(16)	
TRANS_ID_START_STR_ID	INTEGER	STR_ID_DOMAIN
GLOBAL_TID_START	VARCHAR(16)	
GLOBAL_TID_START_STR_ID	INTEGER	STR_ID_DOMAIN
DBS_READS_START	INTEGER	
DBS_WRITES_START	INTEGER	
RUJ_READS_START	INTEGER	
RUJ_WRITES_START	INTEGER	
AIJ_WRITES_START	INTEGER	
ROOT_READS_START	INTEGER	
ROOT_WRITES_START	INTEGER	
BUFFER_READS_START	INTEGER	
GET_VM_BYTES_START	INTEGER	
FREE_VM_BYTES_START	INTEGER	
LOCK_REQS_START	INTEGER	
REQ_NOT_QUEUED_START	INTEGER	
REQ_STALLS_START	INTEGER	
REQ_DEADLOCKS_START	INTEGER	
PROM_DEADLOCKS_START	INTEGER	
LOCK_RELS_START	INTEGER	
LOCK_STALL_TIME_START	INTEGER	
D_FETCH_RET_START	INTEGER	
D_FETCH_UPD_START	INTEGER	
D_LB_ALLOK_START	INTEGER	
D_LB_GBNEEDLOCK_START	INTEGER	
D_LB_NEEDLOCK_START	INTEGER	
D_LB_OLDVER_START	INTEGER	
D_GB_NEEDLOCK_START	INTEGER	
D_GB_OLDVER_START	INTEGER	
D_NOTFOUND_IO_START	INTEGER	
D_NOTFOUND_SYN_START	INTEGER	
S_FETCH_RET_START	INTEGER	
S_FETCH_UPD_START	INTEGER	
S_LB_ALLOK_START	INTEGER	

(continued on next page)

Table 8–9 (Cont.) Columns for Table EPC$1_221_TRANSACTION

Column Name	Data Type	Domain
S_LB_GBNEEDLOCK_START	INTEGER	
S_LB_NEEDLOCK_START	INTEGER	
S_LB_OLDVER_START	INTEGER	
S_GB_NEEDLOCK_START	INTEGER	
S_GB_OLDVER_START	INTEGER	
S_NOTFOUND_IO_START	INTEGER	
S_NOTFOUND_SYN_START	INTEGER	
D_ASYNC_FETCH_START	INTEGER	
S_ASYNC_FETCH_START	INTEGER	
D_ASYNC_READIO_START	INTEGER	
S_ASYNC_READIO_START	INTEGER	
AS_READ_STALL_START	INTEGER	
AS_BATCH_WRITE_START	INTEGER	
AS_WRITE_STALL_START	INTEGER	
AREA_ITEMS_START	VARCHAR(128)	
AREA_ITEMS_START_STR_ID	INTEGER	STR_ID_DOMAIN
BIO_START	INTEGER	
DIO_START	INTEGER	
PAGEFAULTS_START	INTEGER	
PAGEFAULT_IO_START	INTEGER	
CPU_START	INTEGER	
CURRENT_PRIO_START	SMALLINT	
VIRTUAL_SIZE_START	INTEGER	
WS_SIZE_START	INTEGER	
WS_PRIVATE_START	INTEGER	
WS_GLOBAL_START	INTEGER	
CROSS_FAC_2_START	INTEGER	
CROSS_FAC_3_START	INTEGER	
CROSS_FAC_7_START	INTEGER	
CROSS_FAC_14_START	INTEGER	
DBS_READS_END	INTEGER	
DBS_WRITES_END	INTEGER	
RUJ_READS_END	INTEGER	
RUJ_WRITES_END	INTEGER	
AIJ_WRITES_END	INTEGER	
ROOT_READS_END	INTEGER	
ROOT_WRITES_END	INTEGER	
BUFFER_READS_END	INTEGER	

(continued on next page)

Table 8–9 (Cont.) Columns for Table EPC$1_221_TRANSACTION

Column Name	Data Type	Domain
GET_VM_BYTES_END	INTEGER	
FREE_VM_BYTES_END	INTEGER	
LOCK_REQS_END	INTEGER	
REQ_NOT_QUEUED_END	INTEGER	
REQ_STALLS_END	INTEGER	
REQ_DEADLOCKS_END	INTEGER	
PROM_DEADLOCKS_END	INTEGER	
LOCK_RELS_END	INTEGER	
LOCK_STALL_TIME_END	INTEGER	
D_FETCH_RET_END	INTEGER	
D_FETCH_UPD_END	INTEGER	
D_LB_ALLOK_END	INTEGER	
D_LB_GBNEEDLOCK_END	INTEGER	
D_LB_NEEDLOCK_END	INTEGER	
D_LB_OLDVER_END	INTEGER	
D_GB_NEEDLOCK_END	INTEGER	
D_GB_OLDVER_END	INTEGER	
D_NOTFOUND_IO_END	INTEGER	
D_NOTFOUND_SYN_END	INTEGER	
S_FETCH_RET_END	INTEGER	
S_FETCH_UPD_END	INTEGER	
S_LB_ALLOK_END	INTEGER	
S_LB_GBNEEDLOCK_END	INTEGER	
S_LB_NEEDLOCK_END	INTEGER	
S_LB_OLDVER_END	INTEGER	
S_GB_NEEDLOCK_END	INTEGER	
S_GB_OLDVER_END	INTEGER	
S_NOTFOUND_IO_END	INTEGER	
S_NOTFOUND_SYN_END	INTEGER	
D_ASYNC_FETCH_END	INTEGER	
S_ASYNC_FETCH_END	INTEGER	
D_ASYNC_READIO_END	INTEGER	
S_ASYNC_READIO_END	INTEGER	
AS_READ_STALL_END	INTEGER	
AS_BATCH_WRITE_END	INTEGER	
AS_WRITE_STALL_END	INTEGER	
AREA_ITEMS_END	VARCHAR(128)	
AREA_ITEMS_END_STR_ID	INTEGER	STR_ID_DOMAIN

(continued on next page)

Table 8–9 (Cont.) Columns for Table EPC$1_221_TRANSACTION

Column Name	Data Type	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 8–10 shows the REQUEST_BLR table.

Table 8–10 Columns for Table EPC$1_221_REQUEST_BLR

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

8.2 RDO, RDBPRE and RDB$INTERPRET Features

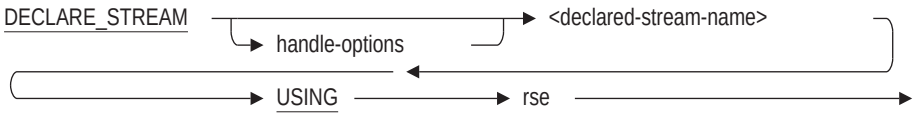
8.2.1 New Request Options for RDO, RDBPRE and RDB$INTERPRET

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.

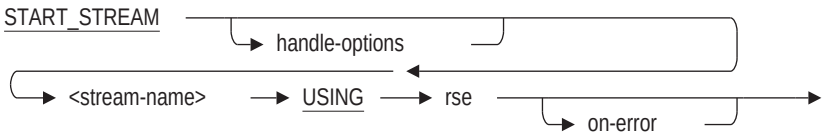
In prior releases, the handle-options could not be specified in interactive RDO or RDB$INTERPRET. This has changed in Rdb but these allowed options will be limited to MODIFY and PROTECTED keywords. For RDBPRE, all options listed will be supported. These option names were chosen to be existing keywords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

The altered statements are shown below.

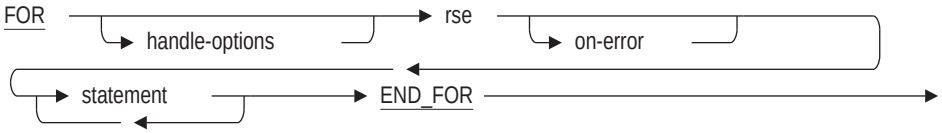
DECLARE_STREAM Format



START_STREAM Format

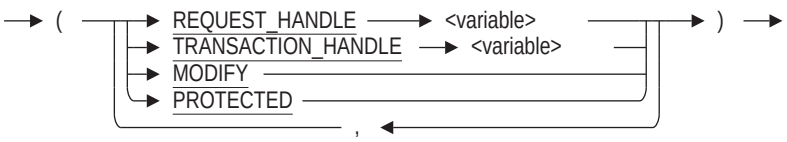


FOR Format



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.

8.2.2 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
   .
   .
   .
```

8.2.3 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 ...

9
Known Problems and Restrictions

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

9.1 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.

9.1.1 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 */ .

$
```

9.1.2 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.

9.1.3 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.

9.1.4 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.

9.1.5 Remote Attach Stalls Before Detecting a Node is Unreachable

Bug 7681548

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

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

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

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

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

9.1.6 Application and Oracle Rdb Both Using SYS$HIBER

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

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

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

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

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

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

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

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

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

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

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

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

9.1.7 Unexpected RCS Termination

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

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

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

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

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

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

This recommendation applies to all versions of Oracle Rdb.

9.1.8 Changes for Processing Existence Logical Names

Oracle Rdb Release 7.2.1.1 changed the handling of so called "existence" logical names used to tune the Rdb environment. These existence logical names could in past versions be defined to any value to enable their effect. The Rdb documentation in most cases described using the value 1 or YES as that value and this change is upward compatible with the documentation.

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

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

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

- RDMS$USE_OLD_COST_MODEL
- RDMS$USE_OLD_COUNT_RELATION
- RDMS$USE_OLD_SEGMENTED_STRING
- RDMS$USE_OLD_UPDATE_RULES

9.1.9 SQL Module or Program Fails with %SQL-F-IGNCASE_BAD

Bug 2351248

A SQL Module or Pre-compiled SQL program built with Rdb 6.1 or earlier may fail when running under Rdb 7.3 if the program submits queries that involve certain kinds of character operations on parameters in the queries. For example, a LIKE operator in the WHERE clause of a SQL statement requires SQL to look for character- or string-matching wildcard characters. Another example is the use of IGNORE CASE which causes SQL to equivalence upper and lower case characters for the character set in use.

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

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

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

If the SQL Module containing the code above is compiled and linked into an executable using a pre-7.0 version of Rdb, it will run properly against that version. However if the same program is run in an Rdb 7.3 or later environment, a call to the SQL_OPN_NAME_LIST procedure will return a SQLCODE of -1. The RDB$MESSAGE_VECTOR will contain a code associated with the following message:

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

To workaround this problem, re-link the program using a 7.3 or later version of SQL$INT.EXE and/or SQL$USER.OLB.

9.1.10 External Routine Images Linked with PTHREAD$RTL

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

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

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

- TO_CHAR
- TO_NUMBER
- TO_DATE

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

9.1.11 Using Databases from Releases Earlier than V7.0

You cannot convert or restore databases earlier than the Oracle Rdb V7.0 format directly to Oracle Rdb V7.3 format. The RMU Convert command for Oracle Rdb V7.3 supports conversions from Oracle Rdb V7.0, V7.1 and V7.2 format databases only. If you have an Oracle Rdb V3.0 through V6.1 format database, you must convert it to at least Oracle Rdb V7.0 format and then convert it to Oracle Rdb V7.3 format. For example, if you have a V4.2 format database, you must convert it first to at least Oracle Rdb V7.0 format, then convert it to Oracle Rdb V7.3 format.

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

9.1.12 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.

9.1.13 New Attributes Saved by RMU/UNLOAD Incompatible With Prior Versions

Bug 2676851

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

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

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

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

9.1.14 SYSTEM-F-INSFMEM Fatal Error With SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED in Galaxy Environment

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

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

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

Galaxy Reboot Required

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

9.1.15 Oracle Rdb and OpenVMS ODS-5 Volumes

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

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

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

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

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

9.1.16 Optimization of Check Constraints

Bug 1448422

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

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

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

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

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

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

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

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

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

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

or:

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

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

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

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

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

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

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

The strategy for a delete on T2 is now:

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

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

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

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

9.1.17 Carryover Locks and NOWAIT Transaction Clarification

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

9.1.18 Unexpected Results Occur During Read-Only Transactions on a Hot Standby Database

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

Because Hot Standby does not write to the snapshot files, the isolation level achieved on the standby database for any read-only transaction is a READ 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.

9.1.19 Row Cache Not Allowed While Hot Standby Replication is Active

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

9.1.21 Control of Sort Work Memory Allocation

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

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

The logical names are:

Table 9–1 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.

9.1.22 The Halloween Problem

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

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

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

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

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

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

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

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

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

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

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

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

9.2 SQL Known Problems and Restrictions

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

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

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

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

This restriction does not apply to interactive or dynamic SQL.

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

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

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

9.2.2 Multistatement or Stored Procedures May Cause Hangs

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

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

Session 1:

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

Session 2:

```
$ RMU/BACKUP/LOG/ONLINE MF_PERSONNEL MF_PERSONNEL
```

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

```
$ RMU/SHOW LOCKS /MODE=BLOCKING MF_PERSONNEL
    .
    .
    .
Resource: nowait signal

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

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

9.2.3 Use of Oracle Rdb from Shareable Images

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

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

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

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

9.3 Oracle RMU Known Problems and Restrictions

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

9.3.1 RMU/CONVERT Fails When Maximum Relation ID is Exceeded

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

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

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

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

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

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

9.3.2 RMU/UNLOAD/AFTER_JOURNAL Requires Accurate AIP Logical Area Information

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

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

EMPLOYEES /TYPE=TABLE

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

EMPLOYEES /AREA=EMPID_OVER /TYPE=TABLE

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

- TABLE

  Specifies that the logical area is a data table. This would be a table created using the SQL CREATE TABLE syntax.

- B-TREE

Specifies that the logical area is a B-tree index. This would be an index created using the SQL CREATE INDEX TYPE IS SORTED syntax.

- HASH

  Specifies that the logical area is a hash index. This would be an index created using the SQL CREATE INDEX TYPE IS HASHED syntax.

- SYSTEM

  Specifies that the logical area is a system record that is used to identify hash buckets. Users cannot explicitly create these types of logical areas.

  Note

  This type should NOT be used for the RDB$SYSTEM logical areas. This type does NOT identify system relations.

- BLOB

  Specifies that the logical area is a BLOB repository.

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

9.3.3 Do Not Use HYPERSORT with RMU/OPTIMIZE/AFTER_JOURNAL Command

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

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

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

9.3.4 Changes in EXCLUDE and INCLUDE Qualifiers for RMU/BACKUP

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

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

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

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

9.3.5 RMU/BACKUP Operations Should Use Only One Type of Tape Drive

When using more than one tape drive for an RMU/BACKUP command, all of the tape drives must be of the same type (for example, all the tape drives must be TA90s or TZ87s or TK50s). Using different tape drive types (for example, one TK50 and one TA90) for a single database backup operation may make database restoration difficult or impossible.

Oracle RMU attempts to prevent using different tape drive densities during a backup operation, but is not able to detect all invalid cases and expects that all tape drives for a backup are of the same type.

As long as all of the tapes used during a backup operation can be read by the same type of tape drive during a restore operation, the backup is likely valid. This may be the case, for example, when using a TA90 and a TA90E.

Oracle Corporation recommends that, on a regular basis, you test your backup and recovery procedures and environment using a test system. You should restore the database and then recover using AIJs to simulate failure recovery of the production system.

Consult the Oracle Rdb7 Guide to Database Maintenance, the Oracle Rdb7 Guide to Database Design and Definition, and the Oracle RMU Reference Manual for additional information about Oracle Rdb backup and restore operations.

9.3.6 RMU/VERIFY Reports PGSPAMENT or PGSPMCLST Errors

RMU/VERIFY may sometimes report PGSPAMENT or PGSPMCLST errors when verifying storage areas. These errors indicate that the Space Area Management (SPAM) page fullness threshold for a particular data page does not match the actual space usage on the data page. For a further discussion of SPAM pages, consult the Oracle Rdb7 Guide to Database Maintenance.

In general, these errors will not cause any adverse affect on the operation of the database. There is potential for space on the data page to not be totally utilized, or for a small amount of extra I/O to be expended when searching for space in which to store new rows. But unless there are many of these errors then the impact should be negligible.

It is possible for these inconsistencies to be introduced by errors in Oracle Rdb. When those cases are discovered, Oracle Rdb is corrected to prevent the introduction of the inconsistencies. It is also possible for these errors to be introduced during the normal operation of Oracle Rdb. The following scenario can leave the SPAM pages inconsistent:

1. A process inserts a row on a page, and updates the threshold entry on the corresponding SPAM page to reflect the new space utilization of the data page. The data page and SPAM pages are not flushed to disk.

2. Another process notifies the first process that it would like to access the SPAM page being held by the process. The first process flushes the SPAM page changes to disk and releases the page. Note that it has not flushed the data page.
3. The first process then terminates abnormally (for example, from the DCL STOP/IDENTIFICATION command). Since that process never flushed the data page to disk, it never wrote the changes to the Recovery Unit Journal (RUJ) file. Since there were no changes in the RUJ file for that data page then the Database Recovery (DBR) process did not need to roll back any changes to the page. The SPAM page retains the threshold update change made above even though the data page was never flushed to disk.

While it would be possible to create mechanisms to ensure that SPAM pages do not become out of synch with their corresponding data pages, the performance impact would not be trivial. Since these errors are relatively rare and the impact is not significant, then the introduction of these errors is considered to be part of the normal operation of Oracle Rdb. If it can be proven that the errors are not due to the scenario above, then Oracle Product Support should be contacted.

PGSPAMENT and PGSPMCLST errors may be corrected by doing any one of the following operations:

- Recreate the database by performing:
  1. SQL EXPORT
  2. SQL DROP DATABASE
  3. SQL IMPORT
- Recreate the database by performing:
  1. RMU/BACKUP
  2. SQL DROP DATABASE
  3. RMU/RESTORE
- Repair the SPAM pages by using the RMU/REPAIR command. Note that the RMU/REPAIR command does not write its changes to an after-image journal (AIJ) file. Therefore, Oracle recommends that a full database backup be performed immediately after using the RMU/REPAIR command.

9.4 Known Problems and Restrictions in All Interfaces for Release 7.0 and Earlier

The following problems and restrictions from release 7.0 and earlier still exist.

9.4.1 Converting Single-File Databases

Because of a substantial increase in the database root file information for V7.0, you should ensure that you have adequate disk space before you use the RMU/CONVERT command with single-file databases and V7.0 or higher.

The size of the database root file of any given database increases a maximum of about 600 disk blocks. The actual increase depends mostly on the maximum number of users specified for the database.

9.4.2 Row Caches and Exclusive Access

If a table has a row-level cache defined for it, the Row Cache Server (RCS) may acquire a shared lock on the table and prevent any other user from acquiring a Protective or Exclusive lock on that table.

9.4.3 Exclusive Access Transactions May Deadlock with RCS Process

If a table is frequently accessed by long running transactions that request READ/WRITE access reserving the table for EXCLUSIVE WRITE and if the table has one or more indexes, you may experience deadlocks between the user process and the Row Cache Server (RCS) process.

There are at least three suggested workarounds to this problem:

- Reserve the table for SHARED WRITE
- Close the database and disable row cache for the duration of the exclusive transaction
- Change the checkpoint interval for the RCS process to a time longer than the time required to complete the batch job and then trigger a checkpoint just before the batch job starts. Set the interval back to a smaller interval after the checkpoint completes.

9.4.4 Strict Partitioning May Scan Extra Partitions

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

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

```
        450   Gentile        Russ
1 row selected
SQL>
```

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

9.4.5 Restriction When Adding Storage Areas with Users Attached to Database

If you try to interactively add a new storage area where the page size is less than the smallest existing page size and the database has been manually opened or users are active, the add operation fails with the following errors:

```
%RDMS-F-NOEUACCESS, unable to acquire exclusive access to database
```

or

```
%RDB-F-SYS_REQUEST, error from system services request
-RDMS-F-FILACCERR, error opening database root DKA0:[RDB]TEST.RDB;1
-SYSTEM-W-ACCONFLICT, file access conflict
```

You can make this change only when no users are attached to the database and, if the database is set to OPEN IS MANUAL, the database is closed. Several internal Oracle Rdb data structures are based on the minimum page size and these structures cannot be resized if users are attached to the database.

Furthermore, because this particular change is not recorded in the AIJ, any recovery scenario fails. Note also that if you use .aij files, you must backup the database and restart after-image journaling because this change invalidates the current AIJ recovery.

9.4.6 Multiblock Page Writes May Require Restore Operation

If a node fails while a multiblock page is being written to disk, the page in the disk becomes inconsistent, and is detected immediately during failover. (Failover is the recovery of an application by restarting it on another computer.) The problem is rare, and occurs because only single-block I/O operations are guaranteed by OpenVMS to be written atomically. This problem has never been reported by any customer and was detected only during stress tests in our labs.

Correct the page by an area-level restore operation. Database integrity is not compromised, but the affected area is not available until the restore operation completes.

A future release of Oracle Rdb will provide a solution that guarantees multiblock atomic write operations. Cluster failovers will automatically cause the recovery of multiblock pages, and no manual intervention will be required.

9.4.7 Replication Option Copy Processes Do Not Process Database Pages Ahead of an Application

When a group of copy processes initiated by the Replication Option (formerly Data Distributor) begins running after an application has begun modifying the database, the copy processes catch up to the application and are not able to process database pages that are logically ahead of the application in the RDB$CHANGES system relation. The copy processes all align waiting for the same database page and do not move on until the application has released it.

The performance of each copy process degrades because it is being paced by the application.

When a copy process completes updates to its respective remote database, it updates the RDB$TRANSFERS system relation and then tries to delete any RDB$CHANGES rows not needed by any transfers. During this process, the RDB$CHANGES table cannot be updated by any application process, holding up any database updates until the deletion process is complete. The application stalls while waiting for the RDB$CHANGES table. The resulting contention for RDB$CHANGES SPAM pages and data pages severely impacts performance throughput, requiring user intervention with normal processing.

This is a known restriction in V4.0 and higher. Oracle Rdb uses page locks as latches. These latches are held only for the duration of an action on the page and not to the end of transaction. The page locks also have blocking asynchronous system traps (ASTs) associated with them. Therefore, whenever a process requests a page lock, the process holding that page lock is sent a blocking AST (BLAST) by OpenVMS. The process that receives such a blocking AST queues the fact that the page lock should be released as soon as possible. However, the page lock cannot be released immediately.

Such work requests to release page locks are handled at verb commit time. An Oracle Rdb verb is an Oracle Rdb query that executes atomically, within a transaction. Therefore, verbs that require the scan of a large table, for example, can be quite long. An updating application does not release page locks until its verb has completed.

The reasons for holding on to the page locks until the end of the verb are fundamental to the database management system.

9.5 SQL Known Problems and Restrictions for Oracle Rdb Release 7.0 and Earlier

The following problems and restrictions from Oracle Rdb Release 7.0 and earlier still exist.

9.5.1 Different Methods of Limiting Returned Rows from Queries

You can establish the query governor for rows returned from a query by using either the SQL SET QUERY LIMIT statement or a logical name. This note describes the differences between the two mechanisms.

If you define the RDMS$BIND_QG_REC_LIMIT logical name to a small value, the query often fails with no rows returned regardless of the value assigned to the logical. The following example demonstrates setting the limit to 10 rows and the resulting failure:

```
$ DEFINE RDMS$BIND_QG_REC_LIMIT 10
$ SQL$
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES;
%RDB-F-EXQUOTA, Oracle Rdb runtime quota exceeded
-RDMS-E-MAXRECLIM, query governor maximum limit of rows has been reached
```

Interactive SQL must load its metadata cache for the table before it can process the SELECT statement. In this example, interactive SQL loads its metadata cache to allow it to check that the column EMPLOYEE_ID really exists for the table. The queries on the Oracle Rdb system relations RDB$RELATIONS and RDB$RELATION_FIELDS exceed the limit of rows.

Oracle Rdb does not prepare the SELECT statement, let alone execute it. Raising the limit to a number less than 100 (the cardinality of EMPLOYEES) but more than the number of columns in EMPLOYEES (that is, the number of rows to read from the RDB$RELATION_FIELDS system relation) is sufficient to read each column definition.

To see an indication of the queries executed against the system relations, define the RDMS$DEBUG_FLAGS logical name as "S" or "B".

If you set the row limit using the SQL SET QUERY statement and run the same query, it returns the number of rows specified by the SQL SET QUERY statement before failing:

```
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SET QUERY LIMIT ROWS 10;
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES;
EMPLOYEE_ID
00164
00165
   .
   .
   .
00173
%RDB-E-EXQUOTA, Oracle Rdb runtime quota exceeded
-RDMS-E-MAXRECLIM, query governor maximum limit of rows has been reached
```

The SET QUERY LIMIT specifies that only user queries be limited to 10 rows. Therefore, the queries used to load the metadata cache are not restricted in any way.

Like the SET QUERY LIMIT statement, the SQL precompiler and module processor command line qualifiers (QUERY_MAX_ROWS and SQLOPTIONS=QUERY_MAX_ROWS) only limit user queries.

Keep the differences in mind when limiting returned rows using the logical name 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.

9.5.2 Suggestions for Optimal Use of SHARED DATA DEFINITION Clause for Parallel Index Creation

The CREATE INDEX process involves the following steps:

1. Process the metadata.
2. Lock the index name.

   Because new metadata (which includes the index name) is not written to disk until the end of the index process, Oracle Rdb must ensure index name uniqueness across the database during this time by taking a special lock on the provided index name.
3. Read the table for sorting by selected index columns and ordering.
4. Sort the key data.
5. Build the index (includes partitioning across storage areas).
6. Write new metadata to disk.

Step 6 is the point of conflict with other index definers because the system relation and indexes are locked like any other updated table.

Multiple users can create indexes on the same table by using the RESERVING table_name FOR SHARED DATA DEFINITION clause of the SET TRANSACTION statement. For optimal usage of this capability, Oracle Rdb suggests the following guidelines:

- You should commit the transaction immediately after the CREATE INDEX statement so that locks on the table are released. This avoids lock conflicts with other index definers and improves overall concurrency.
- By assigning the location of the temporary sort work files SORTWORK0, SORTWORK1, ... , SORTWORK9 to different disks for each parallel process that issues the SHARED DATA DEFINITION statement, you can increase the efficiency of sort operations. This minimizes any possible disk I/O bottlenecks and allows overlap of the SORT read/write cycle.
- If possible, enable global buffers and specify a buffer number large enough to hold a sufficient amount of table data. However, do not define global buffers larger than the available system physical memory. Global buffers allow sharing of database pages and thus result in disk I/O savings. That is, pages are read from disk by one of the processes and then shared by the other index definers for the same table, reducing the I/O load on the table.
- If global buffers are not used, ensure that enough local buffers exist to keep much of the index cached (use the RDM$BIND_BUFFERS logical name or the NUMBER OF BUFFERS IS clause in SQL to change the number of buffers).
- To distribute the disk I/O load, store the storage areas for the indexes on separate disk drives. Note that using the same storage area for multiple indexes results in contention during the index creation (Step 5) for SPAM pages.
- Consider placing the .ruj file for each parallel definer on its own disk or an infrequently used disk.
- Even though snapshot I/O should be minimal, consider disabling snapshots during parallel index creation.
- Refer to the Oracle Rdb7 Guide to Database Performance and Tuning to determine the appropriate working set values for each process to minimize excessive paging activity. In particular, avoid using working set parameters where the difference between WSQUOTA and WSEXTENT is large. The SORT utility uses the difference between these two values to allocate scratch virtual memory. A large difference (that is, the requested virtual memory grossly exceeds the available physical memory) may lead to excessive page faulting.
- The performance benefits of using SHARED DATA DEFINITION can best be observed when creating many indexes in parallel. The benefit is in the average elapsed time, not in CPU or I/O usage. For example, when two indexes are created in parallel using the SHARED DATA DEFINITION clause, the database must be attached twice, and the two attaches each use separate system resources.
- Using the SHARED DATA DEFINITION clause on a single-file database or for indexes defined in the RDB$SYSTEM storage area is not recommended.

The following table displays the elapsed time benefit when creating multiple indexes in parallel with the SHARED DATA DEFINITION clause. The table shows the elapsed time for ten parallel process index creations (Index1, Index2, ... Index10) and one process with ten sequential index creations (All10). In this example, global buffers are enabled and the number of buffers is 500. The longest time for a parallel index creation is Index7 with an elapsed time of 00:02:34.64, compared to creating ten indexes sequentially with an elapsed time of 00:03:26.66. The longest single parallel create index elapsed time is shorter than the elapsed time of creating all ten of the indexes serially.

Table 9–2 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

9.5.3 Side Effect When Calling Stored Routines

When calling a stored routine, you must not use the same routine to calculate argument values by a stored function. For example, if the routine being called is also called by a stored function during the calculation of an argument value, passed arguments to the routine may be incorrect.

The following example shows a stored procedure P being called during the calculation of the arguments for another invocation of the stored procedure P:

```
SQL> create module M
cont>     language SQL
cont>
cont>     procedure P (in :a integer, in :b integer, out :c integer);
cont>     begin
cont>     set :c = :a + :b;
cont>     end;
cont>
cont>     function F () returns integer
cont>     comment is 'expect F to always return 2';
cont>     begin
cont>     declare :b integer;
cont>     call P (1, 1, :b);
cont>     trace 'returning ', :b;
cont>     return :b;
cont>     end;
cont> end module;
SQL>
SQL> set flags 'TRACE';
SQL> begin
cont> declare :cc integer;
cont> call P (2, F(), :cc);
cont> trace 'Expected 4, got ', :cc;
cont> end;
~Xt: returning 2
~Xt: Expected 4, got 3
```

The result as shown above is incorrect. The routine argument values are written to the called routine's parameter area before complex expression values are calculated. These calculations may (as in the example) overwrite previously copied data.

The workaround is to assign the argument expression (in this example calling the stored function F) to a temporary variable and pass this variable as the input for the routine. The following example shows the workaround:

```
SQL> begin
cont> declare :bb, :cc integer;
cont> set :bb = F();
cont> call P (2, :bb, :cc);
cont> trace 'Expected 4, got ', :cc;
cont> end;
~Xt: returning 2
~Xt: Expected 4, got 4
```

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

9.5.4 Considerations When Using Holdable Cursors

If your applications use holdable cursors, be aware that after a COMMIT or ROLLBACK statement is executed, the result set selected by the cursor may not remain stable. That is, rows may be inserted, updated, and deleted by other users because no locks are held on the rows selected by the holdable cursor after a commit or rollback occurs. Moreover, depending on the access strategy, rows not yet fetched may change before Oracle Rdb actually fetches them.

As a result, you may see the following anomalies when using holdable cursors in a concurrent user environment:

- If the access strategy forces Oracle Rdb to take a data snapshot, the data read and cached may be stale by the time the cursor fetches the data.

For example, user 1 opens a cursor and commits the transaction. User 2 deletes rows read by user 1 (this is possible because the read locks are released). It is possible for user 1 to report data now deleted and committed.

- If the access strategy uses indexes that allow duplicates, updates to the duplicates chain may cause rows to be skipped, or even revisited.

  Oracle Rdb keeps track of the dbkey in the duplicate chain pointing to the data that was fetched. However, the duplicates chain could be revised by the time Oracle Rdb returns to using it.

Holdable cursors are a very powerful feature for read-only or predominantly read-only environments. However, in concurrent update environments, the instability of the cursor may not be acceptable. The stability of holdable cursors for update environments will be addressed in future versions of Oracle Rdb.

You can define the logical name RDMS$BIND_HOLD_CURSOR_SNAP to the value 1 to force all hold cursors to fetch the result set into a cached data area. (The cached data area appears as a "Temporary Relation" in the optimizer strategy displayed by the SET FLAGS 'STRATEGY' statement or the RDMS$DEBUG_FLAGS "S" flag.) This logical name helps to stabilize the cursor to some degree.

9.5.5 AIJSERVER Privileges

For security reasons, the AIJSERVER account ("RDMAIJSERVER") is created with only NETMBX and TMPMBX privileges. These privileges are sufficient to start Hot Standby, in most cases.

However, for production Hot Standby systems, these privileges are not adequate to ensure continued replication in all environments and workload situations. Therefore, Oracle recommends that the DBA provide the following additional privileges for the AIJSERVER account:

- ALTPRI - This privilege allows the AIJSERVER to adjust its own priority to ensure adequate quorum (CPU utilization) to prompt message processing.
- PSWAPM - This privilege allows the AIJSERVER to enable and disable process swapping, also necessary to ensure prompt message processing.
- SETPRV - This privilege allows the AIJSERVER to temporarily set any additional privileges it may need to access the standby database or its server processes.
- SYSPRV - This privilege allows the AIJSERVER to access the standby database rootfile, if necessary.
- WORLD - This privilege allows the AIJSERVER to more accurately detect standby database server process failure and handle network failure more reliably.