Oracle® Rdb for OpenVMS

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Oracle® Rdb for OpenVMS

Release Notes

Release 7.2

January 2006

Oracle Rdb Release Notes, Release 7.2 for OpenVMS

Contents

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb Release 7.2. The notes describe changed and enhanced features; upgrade and compatibility information; new and existing software problems and restrictions; and software and documentation corrections.

Intended Audience

This manual is intended for use by all Oracle Rdb users. Read this manual before you install, upgrade, or use Oracle Rdb Release 7.2.

Document Structure

This manual consists of the following chapters:

Chapter 1	Describes how to install Oracle Rdb Release 7.2.
Chapter 2	Describes enhancements introduced in Oracle Rdb Release 7.2.
Chapter 3	Describes problems corrected in Oracle Rdb Release 7.2.
Chapter 4	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb Release 7.2.

Chapter 1
Installing Oracle Rdb Release 7.2

This software update is installed using the OpenVMS VMSINSTAL utility.

NOTE

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

1.1 Oracle Rdb on HP OpenVMS Industry Standard 64

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

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

1.2 Requirements

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

- This Oracle Rdb release requires the following OpenVMS environments:
  - ♦ OpenVMS Alpha V8.2 or later.
  - ♦ OpenVMS Industry Standard 64 V8.2−1 or later.
- Oracle Rdb must be shutdown before you install this update kit. That is, the command file SYS$STARTUP:RMONSTOP72.COM should be executed before proceeding with this installation. If you have an OpenVMS cluster, you must shutdown the Rdb Release 7.2 monitor on all nodes in the cluster before proceeding.
- The installation requires approximately 280,000 blocks for OpenVMS Alpha systems.
- The installation requires approximately 500,000 blocks for OpenVMS I64 systems.
- Oracle strongly recommends that all available OpenVMS patches are installed on all systems prior to installing Oracle Rdb Release 7.2. Contact your HP support representitive for more information and assistance.

1.3 Database Format Changed

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

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

1.4 Using Databases from Releases Earlier than V7.0

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

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

1.5 Deinstall Any Prior PCSI−installed Kit

Users who previously installed the limited−functionality Oracle Rdb 72 ADK kit on I64 systems with the PCSI installation procedure should remove the ADK software prior to installing this full kit. If you have the RDBADK kit installed on your system (you can use the PCSI "PRODUCT SHOW HISTORY RDBADK" command to determine if this software was installed), please use the PCSI "PRODUCT REMOVE RDBADK" command prior to using VMSINSTAL to install the full kit.

If you do not remove the ADK software

If you do not remove the ADK software before using the VMSINSTAL command to install the full 7.2 kit, the installation will fail with the following fatal error:

```
The installation procedure found the existing file,
SYS$COMMON:[SYSLIB]RDMSHR72.EXE (RDB T7.2−100),
to be a higher release version than that supplied with this
kit.  This is a fatal condition and must be checked out before
installation of Oracle Rdb can be attempted again.

You must use RDB$DEINSTALL_DELETE.COM to deinstall the higher
version before installing an older version.
```

Caution

Only use the PCSI "PRODUCT REMOVE RDBADK" command to remove the RDBADK software, do not use the RDB$DEINSTALL_DELETE procedure suggested by the VMSINSTAL procedure.

1.6 Invoking the VMSINSTAL Procedure

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

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

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

```
@SYS$UPDATE:VMSINSTAL RDBV72000IM device−name
```

- To install the Oracle Rdb for OpenVMS Alpha kit that is compiled to run on all Alpha platforms:

```
@SYS$UPDATE:VMSINSTAL RDBV72000AM device−name
```

- To install the Oracle Rdb for OpenVMS Alpha kit that is performance targeted for Alpha EV56 and later platforms:

```
@SYS$UPDATE:VMSINSTAL RDBV72001AM 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]*

1.7 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.

1.8 After Installing Oracle Rdb

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

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

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

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

```
$ LIBRARY /TEXT /EXTRACT=RDBVMSV7.2 −
_$ /OUT=RDBVMS.EPC$DEF SYS$SHARE:EPC$FACILITY.TLB
```

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

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

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

1.9 Maximum OpenVMS Version Check

OpenVMS Version 8.2−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.

1.10 VMS$MEM_RESIDENT_USER Rights Identifier Required

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

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

1.11 Installation, Configuration, Migration, Upgrade Suggestions

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

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

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

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

Table 1−1 Migration Suggestions

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

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

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

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

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

1.12 Sample Installation Dialogue

The following example shows the output from an installation.

```
$ @SYS$UPDATE:VMSINSTAL RDBV72000IM072 DEV:[DIR]
    .
    .
    .
The following products will be processed:

  RDBV72000IM V7.2

        Beginning installation of RDBV72000IM V7.2 at 11:01

%VMSINSTAL-I-RESTORE, Restoring product save set A ...
%VMSINSTAL-I-RELMOVED, Product's release notes have been moved to SYS$HELP.
   Copyright © 1995, 2006, Oracle Corporation.  All Rights Reserved.

    ************************************************************

    SYSTEM MANAGER:

    The RMU Parallel Utilities require SQL/Services V7.0 or later.
    You currently do not have SQL/Services installed on your system.
    Please remember to install SQL/Services before you attempt to
    use this Rdb functionality.
    Please refer to the Oracle Rdb V7.2 Installation Guide and Release
    Notes for further information.

    ************************************************************

        Installation procedure for: "Oracle Rdb V7.2-000"

    You are about to install a multiversion Oracle Rdb kit.

    Be sure you have read the section entitled
        "Preparing Your System and the Installing Account"
    in the installation guide before continuing with the installation.

* Do you want to proceed [YES]?

    Checking system requirements ...


    ************************************************************

    This installation requires the creation of the RDB$REMOTE72
    account.  The installation procedure will not proceed until you
    enter a valid user identification code (UIC) for the RDB$REMOTE72
    account.  The UIC must be unique.  Format [ggg,mmm].

    ************************************************************

* Enter UIC to be used for RDB$REMOTE72 account: [123,100]

    ************************************************************

    This installation requires the creation of the RDMAIJ72
    account.  The installation procedure will not proceed until you
    enter a valid user identification code (UIC) for the RDMAIJ72
```

```
    account.  The UIC must be unique.  Format [ggg,mmm].

    ************************************************************

* Enter UIC to be used for RDMAIJ72 account: [123,101]

    ************************************************************

    This installation requires the creation of the RDMSTT72
    account.  The installation procedure will not proceed until you
    enter a valid user identification code (UIC) for the RDMSTT72
    account.  The UIC must be unique.  Format [ggg,mmm].

    ************************************************************

* Enter UIC to be used for RDMSTT72 account: [123,102]
* Do you want to run the IVP after the installation [YES]?
* Do you want to purge files replaced by this installation [YES]?

    There are no more questions.

    Beginning installation ...

    Installing under VMS V8.2-1 -  1-JAN-2006 21:03

%VMSINSTAL-I-RESTORE, Restoring product save set B ...
%VMSINSTAL-I-RESTORE, Restoring product save set C ...
%VMSINSTAL-I-RESTORE, Restoring product save set D ...
%VMSINSTAL-I-RESTORE, Restoring product save set E ...
%VMSINSTAL-I-SYSDIR, This product creates system disk directory  VMI$ROOT:[SYSHLP.EXAMPLES.RDB72]
%VMSINSTAL-I-SYSDIR, This product creates system disk directory  VMI$ROOT:[SYSTEST.RDB72].

%VMSINSTAL-I-ACCOUNT, This installation creates an ACCOUNT named RDMSTT72.
%UAF-I-ADDMSG, user record successfully added
%UAF-I-RDBADDMSGU, identifier RDMSTT72 value [000123,000502] added to rights database
%VMSINSTAL-I-ACCOUNT, This installation updates an ACCOUNT named RDMSTT72.
%UAF-I-MDFYMSG, user record(s) updated
%VMSINSTAL-I-SYSDIR, This product creates system disk directory  VMI$ROOT:[RDMSTT72].
    ************************************************************

    Oracle Trace has not been installed. Now storing the RDBVMS
    facility definition into SYS$SHARE:EPC$FACILITY.TLB.
    After installing Oracle Trace, the facility definition
    may be placed in the Oracle Trace administration database
    Please refer to the Oracle Trace User's guide for instructions
    on how to insert binary facility definitions into the
    Oracle Trace administration database.

    ************************************************************

    ************************************************************

    SYSTEM MANAGER:

    The following command line MUST be added to the system startup
    command file SYS$STARTUP:SYSTARTUP_VMS.COM for all nodes that will
    be running Oracle Rdb.

        $ @SYS$STARTUP:RMONSTART72

    The following command line should be added to the system
    shutdown command file SYS$MANAGER:SYSHUTDWN.COM for all nodes
```

```
    that will be running Oracle Rdb.

        $ @SYS$MANAGER:RMONSTOP72

    ************************************************************

%VMSINSTAL-I-ACCOUNT, This installation creates an ACCOUNT named RDB$REMOTE72.
%UAF-I-ADDMSG, user record successfully added
%UAF-I-RDBADDMSGU, identifier RDB$REMOTE72 value [000123,000500] added to rights database
%VMSINSTAL-I-ACCOUNT, This installation updates an ACCOUNT named RDB$REMOTE72.
%UAF-I-MDFYMSG, user record(s) updated

    ************************************************************

    SYSTEM MANAGER:

    To have remote access on another node which shares
    this cluster common root directory, you must configure the
    node's DECnet to recognize RDBSERVER.  Do the following:

        a) Log in to that node.
        b) $ @SYS$COMMON:[SYS$STARTUP]RDBSERVER_NCP.COM.

    This command procedure configures RDBSERVER with DECnet on
    that node.  This procedure needs to be executed only ONCE
    per node.

    ************************************************************

%VMSINSTAL-I-SYSDIR, This product creates system disk directory  VMI$ROOT:[RDB$REMOTE72].

%VMSINSTAL-I-ACCOUNT, This installation creates an ACCOUNT named RDMAIJ72.
%UAF-I-ADDMSG, user record successfully added
%UAF-I-RDBADDMSGU, identifier RDMAIJ72 value [000123,000501] added to rights database
%VMSINSTAL-I-ACCOUNT, This installation updates an ACCOUNT named RDMAIJ72.
%UAF-I-MDFYMSG, user record(s) updated
%VMSINSTAL-I-SYSDIR, This product creates system disk directory  VMI$ROOT:[RDMAIJ72].

    ************************************************************

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

    ************************************************************


    ************************************************************

    The Oracle Rdb Installation Verification Procedure (IVP) has
    been provided in SYS$COMMON:[SYSTEST].

    It is invoked using the commands:

      $ SET DEFAULT SYS$COMMON:[SYSTEST]
      $ @RDB$IVP72

    ************************************************************


    ************************************************************

    The release notes for Oracle Rdb are available in the file
```

```
    SYS$HELP:RDB072.RELEASE_NOTES

    ************************************************************

%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target directories...
Current PROCESS Oracle Rdb environment is version V7.2-000 (MULTIVERSION)
Current PROCESS SQL environment is version V7.2-000 (MULTIVERSION)
Current PROCESS Rdb/Dispatch environment is version V7.2-000 (MULTIVERSION)
Oracle Rdb monitor (RDMS_MONITOR72) started

Executing IVP for: Oracle Rdb V7.2-000

Current PROCESS Oracle Rdb environment is version V7.2-000 (MULTIVERSION)
Current PROCESS SQL environment is version V7.2-000 (MULTIVERSION)
Current PROCESS Rdb/Dispatch environment is version V7.2-000 (MULTIVERSION)
 Copyright © 1995, 2006, Oracle Corporation.  All Rights Reserved.

Building the test database.

Beginning Installation Verification Tests.

Running the after-image journaling test.
    Test completed successfully

Running the RDBPRE/BASIC precompiler test.
    Test completed successfully

Running the RDBPRE/COBOL precompiler test.
    Test completed successfully

Running the RDBPRE/FORTRAN precompiler test.
    Test completed successfully

Running the RDML/DEC C preprocessor test.
    Test completed successfully

Running the RDML/PASCAL preprocessor test.
    Test completed successfully

Restoring the SQL database.
    Restore completed successfully

Running the Interactive SQL test.
    Test completed successfully

Running the Dynamic SQL test.
    Test completed successfully

Running the COBOL precompiler test.
    Test completed successfully

Running the FORTRAN precompiler test.
    Test completed successfully

Running the DEC C precompiler test.
    Test completed successfully

Running the PASCAL precompiler test.
    Test completed successfully

Running the SQL MODULE LANGUAGE test for BASIC.
    Test completed successfully
```

```
Running the SQL MODULE LANGUAGE test for C.
    Test completed successfully

IVP completed for: Oracle Rdb V7.2-000

        Installation of RDBV72000IM V7.2 completed at 21:09

    Adding history entry in VMI$ROOT:[SYSUPD]VMSINSTAL.HISTORY

    Creating installation data file: VMI$ROOT:[SYSUPD]RDBV72000IM072.VMI_DATA

        VMSINSTAL procedure done at 21:09
```

Chapter 2
Enhancements And Changes Provided in Oracle Rdb Release 7.2

2.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2

2.1.1 Default Floating Point Format

The Intel Itanium architecture has a 64−bit virtual address model and basic system functions similar to the Alpha architecture. However, there are some implementation differences between the two platforms that might affect user−written applications.

One of the differences is the availability of hardware−supported floating−point formats. The Intel Itanium architecture implements floating−point arithmetic in hardware using the IEEE floating−point formats, including IEEE single and IEEE double. The Alpha architecture supports both IEEE and VAX floating−point formats in hardware, and OpenVMS compilers generate code using the VAX formats by default, with options (on Alpha) to use IEEE formats. Irrespective of whether it was originally written for VAX or Alpha, an OpenVMS application that uses the default VAX floating−point formats needs to produce equivalent behavior on the Intel Itanium architecture using its native IEEE formats.

- On OpenVMS VAX and OpenVMS Alpha, VAX float is the default. VAX format data is assumed and VAX floating instructions are used.
- On OpenVMS Alpha, you can specify the compiler option /FLOAT=IEEE. In this case, IEEE format data is assumed and IEEE floating instructions are used.
- On OpenVMS I64, IEEE float is the default. IEEE format data is assumed and IEEE floating instructions are used.
- On OpenVMS I64, you can specify the compiler option /FLOAT=D_FLOAT or /FLOAT=G_FLOAT.

When you compile an OpenVMS application that specifies an option to use VAX floating−point on the Intel Itanium architecture, the compiler automatically generates code for converting floating−point formats. Whenever the application performs a sequence of arithmetic operations, this code does the following:

1. Converts VAX floating−point formats to either IEEE single or IEEE double floating−point formats.
2. Performs arithmetic operations in IEEE floating−point arithmetic.
3. Converts the resulting data from IEEE formats back to VAX formats.

VAX floating−point formats have the same number of bits and precision as their equivalent IEEE floating−point formats. For most applications, the conversion process will be transparent. Note that where no arithmetic operations are performed (VAX float fetches followed by stores), conversions will not occur. The code handles such situations as moves.

In a few cases, arithmetic calculations might have different results because of the following differences between VAX and IEEE formats:

- Values of numbers represented
- Rounding rules
- Exception behavior

Matching the default of the native IA64 compilers, the Oracle Rdb and SQL precompiler default floating−point format is now IEEE. The default for the Oracle Rdb and SQL precompilers on OpenVMS

Alpha remains as VAX floating−point format. The Oracle Rdb and SQL precompilers on OpenVMS Alpha also support IEEE floating−point format as an option.

For consistent results and data content, it is important that all portions of the application utilize the same floating−point format. Oracle strongly recommends that the floating−point format is explicitly specified on compiler and pre−compiler commands.

For similar behavior for various floating−point exception conditions, Oracle recommends that customers review and consider compiler IEEE floating−point mode options. In particular, the "FAST" option may provide behavior similar to existing applications on VAX and Alpha systems.

The Oracle Rdb on−disk structures and content and data formats remain unchanged in this release.

Oracle recommends reviewing the white paper "OpenVMS Floating−point Arithmetic on the Intel Itanium Architecture" available from HP.

2.1.2 Features Not Yet Available for OpenVMS I64

The following features or capabilities or components are not currently available to run or are known to not run reliably on OpenVMS I64 with this Oracle Rdb release.

- PL/I compiler and Oracle Rdb PL/I precompilers

2.1.3 Expect Additional Memory Consumption

Due to the increased sizes of image files (especially on Integrity servers) and more aggressive buffering and caching schemes and larger I/O size defaults, you should expect to allocate additional page file quota, working set sizes and buffered I/O byte limit quota when using Oracle Rdb Release 7.2.

In particular, when running on Integrity servers, a page file quota of perhaps three times larger may be required for some applications. It is likely that buffered I/O byte limit quota usage may double when moving to Oracle Rdb Release 7.2 (as maximum I/O sizes for some operations are significantly larger than with prior Oracle Rdb Releases).

2.1.4 Handling of Initialized Overlaid Program Sections on OpenVMS I64

On Alpha and VAX systems, initializations can be done to portions of an overlaid program section. Subsequent initializations to the same portions overwrite initializations from previous modules. The last initialization performed on any byte is used as the final one of that byte for the image being linked. On I64 systems, the ELF (Executable and Linkable Format) object language does not implement the feature of the Alpha and VAX object language which allows the initialization of portions of sections. When an initialization is made, the entire section is initialized. Subsequent initializations of this section may be performed only if the non−zero portions match in value.

Any two overlaid sections are compatible if they are identical in the non−zero values. If they are not compatible, the linker issues the following error:

```
%ILINK-E-INVOVRINI, incompatible multiple initializations for overlaid section
section: <section name>
module: <module name for first overlaid section>
file: <file name for first overlaid section>
module: <module name for second overlaid section>
file: <file name for second overlaid section>
```

In the previous message, the linker lists the first module that contributes a non−zero initialization, and the first module with an incompatible initialization. Note that this is not a full list of all incompatible initializations; it is just the first one the linker encounters.

This particular symptom may be seen with applications using Oracle Rdb when multiple modules attempt to initialize handle values. Only one module may initialize any particular handle. SQL precompilers allow initialization to be controlled with the INITIALIZE_HANDLES keyword of the SQLOPTIONS qualifier.

For more detail on the handling of initialized overlaid sections, see the HP OpenVMS Version 8.2 New Features and Documentation Overview.

2.1.5 Deleted Space in Uniform Areas Not Reclaimed by Other Users

Bug 2551066

In prior releases of Oracle Rdb, when rows were deleted from a table stored in a uniform storage area, other database users would not be aware that space was made available and could extend the storage area when inserting additional rows in the table even though free space was available.

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

2.1.6 AIP Entries Cached for Improved Performance

Whenever a table is first accessed within a database attach, Oracle Rdb must look up the description of the table. Some of the table description is stored on disk on pages called Area Inventory Pages (AIPs). These pages are linked together in a special table or "logical area" called RDB$AIP. The AIP pages are sequentially scanned each time it is necessary to find an AIP entry. If a database has many tables defined, then it could take a significant number of I/Os to locate the desired AIP entry in the RDB$AIP list. Prior to this release, the look up was repeated each time a new attach first referenced a table. Thus, applications that often attached to and detached from a database that had many tables defined in it could expend a tremendous amount of I/O constantly reloading AIP entries from disk.

This release introduces an enhancement that, for most applications, should essentially eliminate the RDB$AIP I/O. Now, the first time that a table is referenced the AIP entry is copied into an extended lock value block. (See the OpenVMS Programming Concepts Manual for more information regarding lock value blocks.) Any subsequent reference to the table will find the desired information in the lock value block and thus not need to read the entry from disk. After the most frequently accessed tables have had their AIP entries loaded into lock

value blocks, there should be little, if any, further I/O to the RDB$AIP area.

2.1.7 Improved Rollback Performance

This release of Oracle Rdb introduces additional optimizations for rolling back transactions. These improvements affect the performance of ROLLBACK statements issued by an application and also the database recovery (DBR) process. A summary of the most significant changes are listed below:

- When reading the recovery−unit journal file (RUJ), I/Os are now done using 256 block buffers instead of reading one block at a time as was done in previous versions.
- Multiple buffers are now used to read the journal. While the contents of one buffer are being processed, data is being read into the next buffer asynchronously.
- When writing to the journal, RUJ data is copied directly into the RUJ I/O buffer from the storage area data page instead of being copied into an intermediate buffer and then to the RUJ buffer.
- When reading from the journal, journal entries are processed directly from the RUJ I/O buffer instead of being copied to an intermediate buffer first.
- When rolling back a transaction, the content of the RUJ buffer is scanned to determine what data pages will be rolled back and I/Os are started to those pages immediately. That is, asynchronous prefetches (APF) are issued for pages that will be rolled back. As journal entries are processed, new prefetches are started for subsequent journal entries as soon as buffers are available. This significantly reduces the time spent waiting for I/O completion.
- In previous releases, if a process failed and a DBR was started to recover the user, the DBR would scan the journal to locate the last entry in the journal. For large transactions, the scanning operation could take a considerable amount of time. In this release, the location of the last journal entry is maintained in shared memory. Now, when a DBR process is started it can immediately locate the last entry in the journal without having to scan the journal.

2.1.8 Index Prefetching Performance Improvements

This release of Oracle Rdb introduces an optimization for queries that do index scans to fetch rows from a table. Index scans will now prefetch data pointed to by entries in the index before the application actually requests that the rows be returned. With this optimization, in many instances when an application does request the next row from a result set, the row will already be in an I/O buffer and can be immediately returned to the application.

For example, consider the following table and index definition:

```
CREATE TABLE T1 (C1 INT, C2 INT);
CREATE INDEX I1 ON T1 (C2)
```

The following query will select rows from the table based on a range of values for column C2. Oracle Rdb chooses an index scan retrieval strategy to satisfy the query.

```
SQL> SET FLAGS 'STRATEGY';
SQL> SELECT C1 FROM T1 WHERE C2 > 100 AND C2 < 900000 ORDER BY C2;
  Conjunct        Get     Retrieval by index of relation T1
    Index name  I1 [1:1]
```

When the above query executes, the index node that contains the first C2 value that is greater than 100 is fetched. Then, each entry in the index node that is greater than 100 and less than 900000 is examined and I/O

is started for each data page pointed to by each index entry. Prefetching continues for each entry in the index node until one of the following conditions is met:

- The database ASYNCH PREFETCH DEPTH IS n BUFFERS limit is reached
- The end of the current index node is encountered
- A pointer to a duplicates node is encountered
- The key with the ending scan value (in this example, 900000) is found
- A zig−zag strategy skip is requested

After all possible prefetches have been issued, the first row in the result set is returned to the application. Subsequent fetches for additional rows will find that the I/O request for a needed buffer is already in progress or may even be completed.

Each time that a new entry is requested via the index, if prefetching was stopped due to PREFETCH DEPTH being reached or a new index node being requested, prefetching will resume if that condition is satisfied.

In some applications, the performance improvements from this optimization can be very significant. Large databases that are not readily cached by existing caching products will typically see the greatest improvement in performance.

2.1.9 Performance Improvement for Query with Constant Boolean Predicates

Bug 4205719

The customer reports that the following query where the boolean condition always returns a known value of FALSE uses a full sequential retrieval and becomes very slow on a large table:

```
set flags 'strategy,detail';
select * from resumes where  1 = 2;
Tables:
  0 = RESUMES
Conjunct: 1 = 2
Get     Retrieval sequentially of relation 0:RESUMES
0 row selected
```

Although the condition was always false and 0 rows were returned, Oracle Rdb still performed a sequential table scan. In a database with about 1 million rows, this unnecessary table scan takes a lot of time.

Oracle Rdb has been changed to detect expressions of the following forms and to avoid doing index and table scans if those expressions are non−variable and evaluated as false.

```
WHERE constant−expression
WHERE other−expression AND constant−expression
WHERE constant−expression AND other−expression
```

For example:

```
WHERE 1 = 2
WHERE (1 = 2) AND (LAST_NAME > '')
WHERE (LAST_NAME > '') AND (1 = 2)
```

This does not include expressions that contain host variables, as in the examples below, because host variables are considered to be variable.

```
WHERE :HV = 1
WHERE (:HV1 = 1) AND (LAST_NAME > '')
```

This problem has been corrected in Oracle Rdb Release 7.2.

2.1.10 Index Column Group is Enabled by Default

In prior versions, Index Column Group flag was disabled by default. If this flag is enabled, the Oracle Rdb optimizer will try to find an index that has the same leading columns as the columns of Index Column Group (or Workload Column Group). If a match is found, it uses the index prefix cardinality to calculate the column duplicity and null factors which will help the optimizer to estimate solution costs and cardinalities with higher accuracy.

This flag is now enabled by default.

The following example shows flags that are set by default. This can be overridden using the *RDMS$SET_FLAGS* logical name, or the *SET FLAGS* statement in interactive and dynamic SQL.

```
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),NOBITMAPPED_SCAN
```

RMU Collect Optimizer_Statistics

Oracle strongly recommends that customers use the "RMU /COLLECT OPTIMIZER_STATISTICS" command on databases converted from prior Rdb versions.

2.1.11 No File−System Caching When Writing Database and AIJ Backup Files

It is expected that the disk−based output file from a database or after−image journal backup operation may be relatively large, sequentially accessed and not read in the near future. In order to avoid "polluting" the file system cache and to streamline file write operations, caching by the operating system is now explicitly disabled when writing these files.

2.1.12 Estimation Refinement Rules are Enabled by Default

In prior versions, index estimation was normally performed by descending to the split level in sorted indexes. For more information, please refer to the technical article entitled "Guide to Database Performance and Tuning: Predicate Estimation".

Estimation refinement rules were available to enable greater precision in estimation on indexes of *TYPE IS SORTED RANKED* and to enable estimation on hashed indexes. These rules were enabled using the

REFINE_ESTIMATES flag.

This flag is now enabled by default so that all estimation refinement rules are enabled.

The following example shows flags that are set by default. This can be overridden using the *RDMS$SET_FLAGS* logical name, or the *SET FLAGS* statement in interactive and dynamic SQL.

```
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(127),NOBITMAPPED_SCAN
```

Notice that the *REFINE_ESTIMATES* flag has a value of 127. Please refer to the technical article above for information on the significance of this value.

The previous behavior can be obtained by setting this flag to zero or negating the flag to disable all refinement rules.

```
SQL> set flags 'refine_estimates(0)'
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(0),NOBITMAPPED_SCAN
SQL> exit
$ define rdms$set_flags "refine_estimates(0)"
$ sql
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(0),NOBITMAPPED_SCAN
SQL>exit
$ define rdms$set_flags "norefine_estimates"
$ sql
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(0),NOBITMAPPED_SCAN
SQL>exit
$ sql
SQL> set flags 'norefine_estimates'
SQL> show flags

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(0),NOBITMAPPED_SCAN
```

2.1.13 New LIMIT TO Syntax

This release of Oracle Rdb enhances the LIMIT TO clause by allowing the programmer to skip over some number of delivered rows from the query. For instance, the first row in the result set might be the column headings loaded from a CSV data source loaded by RMU/LOAD/RECORD=FORMAT=DELIMITED and as such should be ignored by queries.

Note

Oracle recommends that the values specified for skip−expression be kept small for performance reasons. These skipped rows are still fetched and processed by the query, they are just not returned to the application.

The following example shows one use of this new feature. This query returns the 100th employee from the EMPLOYEES table.

```
SQL> select last_name, first_name, employee_id
cont> from employees
cont> order by employee_id
cont> limit to 1 skip 99 rows;
 LAST_NAME        FIRST_NAME   EMPLOYEE_ID
 Herbener         James        00471
1 row selected
```

To retrieve the last row in the sorted list, the application programmer could replace the literal value with a subselect that calculates the value. This query also shows the output from the SET FLAGS command for the query strategy. Note the "Skipn" keyword that describes the new SKIP clause.

```
SQL> set flags 'strategy,detail';
SQL> select last_name, first_name, employee_id
cont> from employees
cont> order by employee_id
cont> limit to 1
cont> skip (select count(*)−1 from employees) rows;
Tables:
  0 = EMPLOYEES
  1 = EMPLOYEES
Cross block of 2 entries
  Cross block entry 1
    Aggregate: 0:COUNT (*)
    Index only retrieval of relation 1:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Cross block entry 2
    Firstn: 1
    Skipn: <agg0> − 1
    Get     Retrieval by index of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
 LAST_NAME        FIRST_NAME   EMPLOYEE_ID
 Herbener         James        00471
1 row selected
SQL>
```

An alternative to this query would be to use ORDER ... DESC and then to use a simple LIMIT 1 ROW clause.

This query finds the statistical median salary.

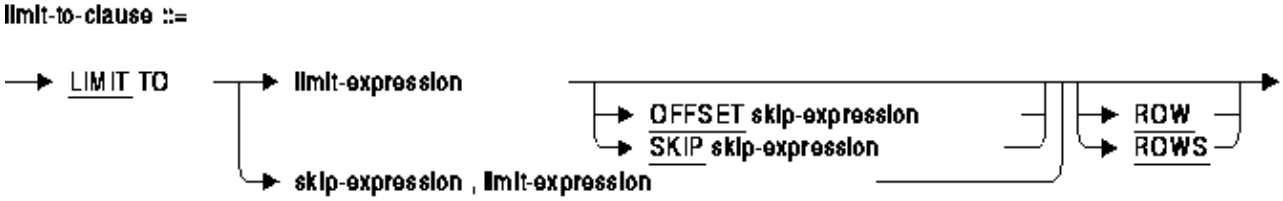
```
SQL> -- select the median salary
SQL> select salary_amount
cont> from salary_history
cont> where salary_end is NULL
cont> order by salary_amount
cont> limit to 1
cont> skip (select count(*)/2
cont>       from salary_history
cont>       where salary_end is NULL);
 SALARY_AMOUNT
    $24,166.00
1 row selected
SQL>
```

The statistical median salary can be compared with the average salary.

```
SQL> -- select the median salary compare with average
SQL> select salary_amount as median_salary,
cont>        (select avg (salary_amount)
cont>         from salary_history
cont>         where salary_end is NULL) as avg_salary edit using SALARY
cont> from salary_history
cont> where salary_end is NULL
cont> order by salary_amount
cont> limit to 1
cont> skip (select count(*)/2
cont>       from salary_history
cont>       where salary_end is NULL);
 MEDIAN_SALARY    AVG_SALARY
    $24,166.00    $31,922.79
1 row selected
SQL>
```

Syntax

The revised SQL syntax for the LIMIT TO clause is:



The syntax variants are supported by different SQL implementations and permit different tools to use this syntax with Oracle Rdb.

Arguments

- limit−expression
  If limit−expression is evaluated to a negative value or zero then no rows are returned from the query and no error is reported.
- skip−expression
  If skip−expression is evaluated to a negative value or zero then no rows are skipped. If the

skip−expression is larger than the rows in the result set then no rows are returned from the query and no error is reported.

If either limit−expression or skip−expression is specified as a numeric literal then it must be an unscaled value. These numeric expressions are converted to BIGINT before executing the query.

Neither limit−expression nor skip−expression may reference columns from the select−expression in which they occur. Only columns of a subselect specified for the limit−expression or skip−expression can be used. See above for examples that use a subselect in the LIMIT TO clause.

2.1.14 Additional %CDD−I−BLRSYNINFO Integrating with CDD

When an Oracle Rdb Release 7.2 database is integrated into a Common Data Dictionary repository, an additional %CDD−I−BLRSYNINFO is generated compared to Oracle Rdb Release 7.1. This additional message is caused by an enhancement to the RDB$DATABASE metadata table so that the value of the RDB$FILE_NAME column can be computed using a function call. The additional informational message is expected and can be ignored.

For example, executing the following SQL statements with Oracle Rdb Release 7.1 would result in a single instance of the %CDD−I−BLRSYNINFO message after the INTEGRATE statement but under Oracle Rdb Release 7.2, it results in two %CDD−I−BLRSYNINFO messages as shown.

```
SQL> CREATE DATABASE FILENAME TEMP;
SQL> DISCONNECT ALL;
SQL> INTEGRATE DATABASE FILENAME TEMP CREATE PATHNAME TEMP;
%CDD−I−BLRSYNINFO, unsupported entity − marked Incomplete
%CDD−I−BLRSYNINFO, unsupported entity − marked Incomplete
```

2.1.15 RMU Unload Record_Definition Accepts TRIM Option

This release of Oracle Rdb adds a TRIM option to the RMU Unload Record_Definition qualifier. The new TRIM option supports three keywords:

- TRAILING − trailing spaces will be trimmed from CHARACTER and CHARACTER VARYING (VARCHAR) data that is unloaded. This is the default setting if only the TRIM option is specified.
- LEADING − leading spaces will be trimmed from CHARACTER and CHARACTER VARYING (VARCHAR) data that is unloaded.
- BOTH − both leading and trailing spaces will be trimmed.

The following example shows the output without using the TRIM option.

```
$ RMU/UNLOAD/RECORD=(FORMAT=DELIMITED) DB$ WORK_STATUS SYS$OUTPUT:
"0","INACTIVE","RECORD EXPIRED"
"1","ACTIVE  ","FULL TIME     "
"2","ACTIVE  ","PART TIME     "
%RMU−I−DATRECUNL,   3 data records unloaded.
```

The results, after adding the TRIM=BOTH option, show that all trailing spaces are removed.

```
$ RMU/UNLOAD/RECORD=(FORMAT=DELIMITED,TRIM=BOTH) DB$ WORK_STATUS SYS$OUTPUT:
"0","INACTIVE","RECORD EXPIRED"
"1","ACTIVE","FULL TIME"
"2","ACTIVE","PART TIME"
%RMU-I-DATRECUNL,   3 data records unloaded.
```

2.1.16 Maximum Page and Buffer Size Increases

Previously, the maximum allowed database buffer size was 64 blocks and the maximum allowed database page size was 32 blocks. These limits have been increased. The current maximum allowed database buffer size is 128 blocks and the maximum allowed database page size is 63 blocks.

Be aware that using larger database buffer sizes will require additional virtual memory and buffered I/O byte count quota.

2.1.17 Various I/O Sizes Increased

Previously, nearly all Oracle Rdb related I/O requests were limited to a maximum of 127 blocks (65,024 bytes). In many areas within Oracle Rdb, this limit has been increased to 256 blocks (131,072 bytes). Writing to and reading from SORT work files is now done with I/O requests up to 1024 blocks (524,288 bytes). These increases should, in some cases, serve to reduce I/O counts, and increase effective I/O through−put at a lower CPU cost (by doing fewer, large I/Os).

This change also will increase the amount of virtual and physical memory required for processes in terms of page file quota and buffered I/O byte count limit.

2.1.18 New Statistics for the Oracle Rdb Executive

Bug 3917094

Several new statistics have been added to a new statistics screen for *RMU/SHOW STATISTICS*.

The following example shows the new screen.

```
Rate: 3.00 Seconds          Rdb Executive Statistics      Elapsed: 00:03:05.26
Page: 1 of 1           DUA0:[PERS.V72]MF_PERSONNEL.RDB;1         Mode: Online
------------------------------------------------------------------------------
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
queries compiled              18       18        0.3           60          60.0
index scans                   13       13        0.2           42          42.0
 index only                    5        5        0.0           17          17.0
 index full                    5        5        0.0           17          17.0
dynamic optimizer              5        5        0.0           17          17.0
 one abandoned                 0        0        0.0            0           0.0
 all abandoned                 0        0        0.0            0           0.0
```

In this screen, the statistics displayed have the following meanings.

- The "queries compiled" statistic counts the number of times the executive has compiled a request, also called query optimization.

- The "index scans" statistic counts the number of times an index scan is initiated. This statistic accumulates for all index types and for all scan types including direct lookup.
- The "index only" statistic counts the number of scans that are started that are index only scans.
- The "index full" statistic counts the number of index scans started that do not have a lower or upper bound. In this case, the entire index will be scanned. If a key value range is provided by the query that includes all keys in the index, the entire index will be scanned. However, this statistic will not be incremented.
- The "dynamic optimizer" statistic counts the number of times the dynamic optimizer has been invoked.
- The "one abandoned" statistic counts the number of times that the dynamic optimizer abandons a background index scan because it is considered too costly.
- The "all abandoned" statistic counts the number of times that all background indexes have been abandoned and the dynamic optimizer has switched to sequential retrieval.

2.1.19 RMU /SHOW STATISTICS Enhanced Navigation Between Row Caches

Bugs 4727723 and 3738511

Previously, when using the RMU /SHOW STATISTICS "Row Cache Utilization", "Hot Row Information", "Row Cache Status", "Row Cache Queue Length", and "Row Length Distribution" displays, it was difficult to move between displays for multiple row caches.

This problem has been corrected. The "]" and "[" keys can be used to navigate next or previous between row caches on these displays.

2.1.20 RMU SHOW LOCKS /RESOURCE_TYPE Qualifier

Previously, the RMU /SHOW LOCKS command would display all lock resource types. This could sometimes result in a significant amount of output and could make it cumbersome to locate locks for specific types of resources.

This situation has been improved with the /RESOURCE_TYPE=(restyp...) qualifier. When this qualifier is present on the command line, only the specific resource types will be displayed. This permits, for example, only PAGE or RECORD lock types to be selected. This functionality is intended primarily as a debugging tool. Knowledge of the lock types and functionality of Oracle Rdb is assumed. Not all lock types will exist on all systems and versions of Oracle Rdb.

The following keywords are allowed with the /RESOURCE_TYPE qualifier.

Table 2−1 RESOURCE_TYPE keywords

Internal Lock Type Name	Keyword(s)
ACCESS	ACCESS
ACTIVE	ACTIVE
AIJDB	AIJDB

AIJFB	AIJFB
AIJHWM	AIJHWM, AIJ_HIGH_WATER_MARK
AIJLOGMSG	AIJ_LOG_MESSAGE
AIJLOGSHIP	AIJ_LOG_SHIPPING
AIJOPEN	AIJ_OPEN
AIJSWITCH	AIJ_SWITCH
AIJ	AIJ
AIPQHD	AIP
ALS	ALS_ACTIVATION
BCKAIJ	AIJ_BACKUP, BCKAIJ
BCKAIJ_SPD	AIJ_BACKUP_SUSPEND
BUGCHK	BUGCHECK
CHAN	CHAN, FILE_CHANNEL
CLIENT	CLIENT
CLOSE	CLOSE
CLTSEQ	CLTSEQ
CPT	CORRUPT_PAGE_TABLE, CPT
DASHBOARD	DASHBOARD_NOTIFY
DBK_SCOPE	DBKEY_SCOPE
DBR	DBR_SERIALIZATION
DB	DATABASE
FIB	FAST_INCREMENTAL_BACKUP, FIB
FILID	FILID
FRZ	FREEZE
GBL_CKPT	GLOBAL_CHECKPOINT
GBPT_SLOT	GLOBAL_BPT_SLOT
KROOT	KROOT
LAREA	LAREA, LOGICAL_AREA
LOGFIL	LOGFIL
MEMBIT	MEMBIT
MONID	MONID, MONITOR_ID
MONITOR	MONITOR
NOWAIT	NOWAIT
PLN	DBKEY, RECORD, PLN
PNO	PAGE, PNO
QUIET	QUIET
RCACHE	RCACHE
RCSREQUEST	RCS_REQUEST
RCSWAITRQST	RCS_WAIT_REQUEST
REL_AREAS	RELEASE_AREAS
REL_GRIC_REQST	RELEASE_GRIC_REQUEST
RMUCLIENT	RMU_CLIENT

ROOT_AREA	DUMMY_ROOT_AREA
RO_L1	L1_SNAP_TRUNCATION
RTUPB	RTUPB
RUJBLK	RUJBLK
RW_L2	L2_SNAP_TRUNCATION
SAC	SNAP_AREA_CURSOR
SEQBLK	SEQBLK
STAREA	STORAGE_AREA, PAREA
STATRQST	STATISTICS_REQUEST
TRM	TERMINATION
TSNBLK	TSNBLK
UTILITY	UTILITY

The RESOURCE_TYPE qualifier is incompatible with the MODE, LIMIT, LOCK and PROCESS qualifiers.

2.1.21 RMU Command Qualifiers Accept Absolute or Delta Date/Time Specification

The allowed date/time format for several RMU command qualifiers have been extended to include delta as well as absolute times. The following command qualifiers now allow delta or absolute time specifications:

- RMU /SHOW STATISTICS /UNTIL=date/time
- RMU /UNLOAD /AFTER_JOURNAL /BEFORE=date/time
- RMU /UNLOAD /AFTER_JOURNAL /SINCE=date/time
- RMU /CHECKPOINT /UNTIL=date/time
- RMU /BACKUP /TAPE_EXPIRATION=date/time
- RMU /BACKUP /AFTER_JOURNAL /TAPE_EXPIRATION=date/time
- RMU /OPTIMIZE /AFTER_JOURNAL /TAPE_EXPIRATION=date/time
- RMU /BACKUP /AFTER_JOURNAL /UNTIL=date/time
- RMU /RECOVER /UNTIL=date/time
- RMU /DUMP /AFTER_JOURNAL /FIRST=TIME=date/time
- RMU /DUMP /AFTER_JOURNAL /LAST=TIME=date/time

Absolute time includes a specific date or time of day. An absolute date/time has one of the following formats:

- dd−mmm−yyyy
- hh:mm:ss.cc
- dd−mmm−yyyy:hh:mm:ss.cc
- "dd−mmm−yyyy hh:mm:ss.cc"
- BOOT
- LOGIN
- TODAY
- TOMORROW
- YESTERDAY

You can omit any of the trailing fields in the date or time. You can omit any of the fields in the middle of the format as long as you specify the punctuation marks, for example, "−mmm−yyyy hh".

Delta time is an offset from the current time to a time in the future. Delta time has the following format:

- "+[dddd−][hh:mm:ss.cc]"

You can truncate delta time after the hour field. You can also omit any of the fields after the hour field format as long as you specify the punctuation marks.

2.1.22 64−bit Statistics

In prior versions of Oracle Rdb, statistics counters were maintained in 32−bit longword integers. This limited counters to a maximum value of 4,294,967,294. This limit could be exceeded and would cause counters to "wrap" back to zero.

This problem has been corrected. Oracle Rdb statistics counters have been promoted to 64−bit quadword integers. This change effects the binary statistics output file format as well.

Note, however, that most field displays within the RMU /SHOW STATISTICS utility have not been widened and may overflow if the internal counter value exceeds the decimal display width.

2.1.23 Maximum Global Buffer Count Increased

Enhancement Bug 3820284

Prior versions of Oracle Rdb limited the total number of global buffers per database to 524,288. This limit has been relaxed. The maximum global buffer count allowed for Oracle Rdb Release 7.2 is 1,048,576.

2.1.24 Support for WORM (Write Once Read Many) Storage Removed

Prior versions of Oracle Rdb provided support for write−once storage areas on write−once, read−many (WORM) optical disk devices. This support has been removed in Oracle Rdb Release 7.2.

Databases containing storage areas configured as "WRITE ONCE" or "WORM" may not be converted to Oracle Rdb Release 7.2 format nor may they be restored using Oracle Rdb Release 7.2.

The various "WRITE ONCE" or "WORM" keywords and qualifiers in RDO, RMU and SQL are deprecated.

The SQL deprecated message is "%SQL−I−DEPR_FEATURE, Deprecated Feature: WRITE ONCE no longer supported − assuming READ WRITE attribute".

The deprecated message is generated when one specifies a clause:

- WRITE ONCE
- WRITE ONCE (JOURNAL IS ENABLED)
- WRITE ONCE (JOURNAL IS DISABLED)

within one of the following statements:

- ALTER DATABASE ... ADD STORAGE AREA

- ALTER DATABASE ... ALTER STORAGE AREA
- CREATE DATABASE ... CREATE STORAGE AREA
- IMPORT DATABASE ... CREATE STORAGE AREA

For example:

```
SQL> CREATE DATA FILENAME WORM_TEST
cont> CREATE STORAGE AREA WORM_AREA WRITE ONCE;
%SQL-I-DEPR_FEATURE, Deprecated Feature: WRITE ONCE no
longer supported - assuming READ WRITE attribute
```

Use of read−only media continues to be available. A database or storage area may be marked read−only and moved to optical media and accessed in a read−only fashion.

2.1.25 Support for ACE (AIJ Cache on Electronic disk) Removed

Prior versions of Oracle Rdb provided support for a file called an AIJ cache on an electronic disk (also known as ACE) to use as a temporary cache for AIJ write operations. At one point in time, these sorts of devices provided a performance benefit for some classes of applications that heavily used the after−image journal.

With changes in technologies (in particular, improved I/O interfaces and various write−back caching schemes), the benefits of the ACE feature have declined to the point where it is no longer an effective performance advantage. Therefore this support has been removed in Oracle Rdb Release 7.2.

The database attribute "CACHE FILENAME ..." is now ignored by Oracle Rdb. The various related keywords and qualifiers in RMU, RDO and SQL are deprecated.

2.1.26 RMU Support for /DENSITY = SDLT320

Oracle Rdb RMU commands that support the /DENSITY qualifier (ie, RMU/BACKUP, RMU/BACKUP/AFTER_JOURNAL and RMU/OPTIMIZE_AIJ) now support the keyword "SDLT320" for use with SuperDLT320 tape drives.

2.1.27 Sequential Scan Statistics

Bug 3917080

Previously, there was no way to accurately determine the number of strict sequential scans nor the number of DBKEYs returned from those sequential scans.

This problem has been corrected in Oracle Rdb Release 7.2. Two new statistics counters record the number of sequential scans started and the number of DBKEYs returned from those sequential scans. These counters are recorded on a database−wide basis (displayed on the "Record Statistics" screen) and on a per−table basis (displayed on the "Logical Area Statistics" screens).

2.1.28 RDB$SHOVER, RDB$SETVER, SQL$SETVER Temporary Files

Previously, the RDB$SHOVER.COM, RDB$SETVER.COM and SQL$SETVER.COM procedures created temporary files when determining image file identifications. This file creation and deletion activity placed undue burden on the system and also restricted the speed of the procedures.

This problem has been corrected in Oracle Rdb Release 7.2. These procedures no longer create and delete temporary files.

2.1.29 Logical RDM$BIND_RW_TX_CHECKPOINT_ADVANCE Removed

Bug 1584167

Prior to Release 7.1.2, if the logical RDM$BIND_RW_TX_CHECKPOINT_ADVANCE was not defined to be 1, read write transactions that did not make any database modifications would not advance their fast commit checkpoint location. In Release 7.1.2, in response to Bug 2439694, the checkpointing code was restructured such that checkpoints may advance at the end of any transaction, whether or not the transactions made any database modifications. That change made the logical RDM$BIND_RW_TX_CHECKPOINT_ADVANCE no longer necessary. It has been removed.

2.1.30 Backup File Encryption

Oracle Rdb supports encryption of .RBF backup files and .AIJ after−image journal backup files using the new /ENCRYPT qualifier.

Encryption can help increase the level of security on backup data that leaves your security domain or premises. To provide a higher level of security, the backup files are always encrypted with a unique internal key. Even though you may use the same RMU command backing up the same data, the encrypted file differs from the previous backup. This is transparent to the user and the same key is used to decrypt the data.

This feature uses the OpenVMS ENCRYPT component which is included with the operating system starting with OpenVMS V8.2. All encryption algorithms supported by OpenVMS ENCRYPT can be used with RMU. Review the online help and the ENCRYPT documentation for details and supported encryption algorithms. The OpenVMS ENCRYPT component must be installed prior to using the /ENCRYPT qualifier with RMU commands.

Encryption Messages

In order to get the correct message text for encryption messages when running RMU/ENCRYPT, the following file needs to be installed using this command:

```
$INSTALL ADD SYS$MESSAGE:ENCRYPT$_MSG.EXE/OPEN/SHARED
```

The process of encryption takes readable data, called plaintext, and uses a mathematical algorithm to transform the plaintext into an unreadable, unintelligible form, called ciphertext.

To encrypt the plaintext data, the encryption operation requires a key. The key is a variable that controls the encryption operation. The same plaintext, encrypted with different keys, results in different ciphertext. In addition, repeated encryption of the same plaintext with the same key also results in different ciphertext each time.

To gain access to the data in an encrypted file, reverse the encryption process by performing the decryption process. Decryption uses a mathematical encryption algorithm to change ciphertext into the original plaintext.

You can either specify an encryption key directly or predefine a key with DCL−ENCRYPT and use the key name instead in the RMU command line.

Encryption Key

If you cannot remember the encryption key you have effectively lost all data in the encrypted file.

2.1.30.1 Commands Accepting /ENCRYPT

The "/ENCRYPT" qualifier is available for the following commands:

- RMU/BACKUP
- RMU/RESTORE
- RMU/RECOVER
- RMU/DUMP/BACKUP
- RMU/BACKUP/AFTER_JOURNAL
- RMU/DUMP/AFTER_JOURNAL
- RMU/OPTIMIZE/AFTER_JOURNAL

FORMAT=NEW_TAPE

After−image journal backup files have to be in the new tape format (/FORMAT=NEW_TAPE) in order to specify /ENCRYPT.

The /ENCRYPT qualifier has the following format: *Encrypt=([Value=|Name=][,Algorithm=])*

Table 2−2 Encrypt Keywords

Keyword	Description
NAME=key−name	Required if you do not specify key−value. Existing key name previously created and stored in the key storage table with the ENCRYPT /CREATE_KEY command. Specify either the name or the value of a key, but not both.
VALUE=key−value	Required if you do not specify key−name. Interactively defines a value for the key. Specify one of the following: • Character string enclosed in quotation marks (""). • 1 to 243 alphanumeric characters. Dollar signs and underscores are

	valid. Hexadecimal constant using the digits 0 to 9 and A to F. Specify either the name or the value of a key, but not both.
ALGORITHM=DESCBC | DESECB | DESCFB	Algorithm used to encrypt the initialization vector and the key you supply. DESCBC is the default.

Specify a key value as a string or the name of a predefined key that was created with the ENCRYPT /CREATE_KEY command. If no algorithm name is specified, the default is DESCBC. For details on the Value, Name and Algorithm parameters, review the "Encryption for OpenVMS Installation and Reference Manual".

2.1.30.2 Examples

- The following example creates a backup file which is encrypted with the specified key value string and the default encryption algorithm.

```
$ RMU/BACKUP/ENCRYPT=(VALUE="My secret key") -
    MYDB.RDB MYBACKUP.RBF
```

  This backup would be restored using a command similar to this example:

```
$ RMU/RESTORE/ENCRYPT=(VALUE="My secret key") -
    MYBACKUP.RBF
```

- The following example creates a backup file which is encrypted with the specified key name and the default encryption algorithm.

```
$ ENCRYPT /CREATE_KEY /LOG HAMLET -
    "And you yourself shall keep the key of it"
%ENCRYPT-S-KEYDEF, key defined for key name = HAMLET
$ RMU/BACKUP/ENCRYPT=NAME=HAMLET MYDB.RDB MYBACKUP.RBF
```

  This backup would be restored using a command similar to this example:

```
$ RMU/RESTORE/ENCRYPT=NAME=HAMLET MYBACKUP.RBF
```

2.1.31 RMU /POPULATE_CACHE Command /[NO]ONLY_CACHED Qualifier

The RMU /POPULATE_CACHE command allows one or more tables and indexes to be read from the database and stored in caches (if they exist). A new qualifier /[NO]ONLY_CACHED can be used to indicate that all specified tables or indexes are to be read or only those with an associated row cache.

Table 2−3 describes the command qualifiers for the RMU /POPULATE_CACHE command.

Table 2−3 RMU /POPULATE_CACHE Command Qualifiers

Qualifier	Description
/TABLE=table−list	Specifies names of one or more tables to fetch. All rows are fetched from each table. If you list multiple tables, separate the table names with a comma, and enclose the list within parentheses. Wildcard characters "*" and "%" are allowed.
/INDEX=index−list	Specifies names of one or more indexes to fetch. All nodes are fetched from each index. If you list multiple indexes, separate the index names with a comma, and enclose the list within parentheses. Wildcard characters "*" and "%" are allowed.
/LOG	Specifies whether the processing of the command is reported to SYS$OUTPUT. Specify the Log qualifier to request that information about the operation be displayed. If you specify neither /NOLOG nor /LOG, the default is the current setting of the DCL verify switch. (The DCL SET VERIFY command controls the DCL verify switch.)
/[NO]ONLY_CACHED	Specifies if table or index content is to be read only if the table or index has an associated row cache. The default is to read data only from objects that have a cache. If /NOONLY_CACHED is specified, then all data from the specified tables or indexes is read.
/TRANSACTION_TYPE=transaction_mode	Allows you to specify the transaction mode, isolation level, and wait behavior for transactions. Use one of the following keywords to control the transaction mode: • AUTOMATIC − When Transaction_Type=Automatic is specified, the transaction type depends on the current database settings for snapshots (enabled, deferred, or disabled), transaction modes available to this user, and the standby status of the database. Automatic mode is the default. • READ_ONLY − Starts read−only transactions. • WRITE − Starts read−write transactions.

2.1.32 RMU/SHOW LOCKS Includes Time and Node Name

Bug 4761828

The output of the RMU/SHOW LOCKS command has been enhanced to include the current date and time and the system node name in the header line as shown in the following example:

```
$ RMU /SHOW LOCKS
================================================================
  SHOW LOCKS Information at 26−NOV−2005 09:29:01.21 on node RDBI64
================================================================

----------------------------------------------------------------
Resource Name: AIJ journal control
Granted Lock Count: 7,  Parent Lock ID: 180007FA,   Lock Access Mode:
Executive, Resource Type: Global, Lock Value Block: 00000013 00000000
00000000 00000000
```

.
.
.

2.1.33 Default /ROW_COUNT Increased for RMU/UNLOAD and RMU/LOAD

The default value for the /ROW_COUNT qualifier for the RMU/LOAD and RMU/UNLOAD commands has been increased from 50 to 500.

The /ROW_COUNT qualifier specifies that Oracle Rdb buffer multiple rows between the Oracle Rdb server and the RMU Load process. The default is 500 rows; however, this value should be adjusted based on working set size and length of loaded data. Increasing the row count may reduce the CPU cost of the load operation. For remote databases, this may significantly reduce network traffic for large volumes of data because the buffered data can be packaged into larger network packets.

The minimum value you can specify for n is 1. The default row size is the value specified for the Commit_Every qualifier or 500, whichever is smaller.

Chapter 3
Software Errors Fixed in Oracle Rdb Release 7.2

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 Various Sequence Generation Problems Fixed

Several problems with sequence value generation have been corrected in this release. These problems occur at the extreme end of value ranges.

1. In some cases, when the end of the range of values is reached and the remaining values do not fill a cache, Rdb would discard the remaining cached values so that the last few values were never used. The cache size is defined by the CACHE clause of CREATE/ALTER SEQUENCE, or the SET FLAGS 'SEQ_CACHE(n)' statement. The following shows an example of this issue.

```
SQL> show sequence s3;
     S3
 Sequence Id: 1
 Initial Value: 29
 Minimum Value: 20
 Maximum Value: 30
 Next Sequence Value: 29
 Increment by: 1
 Cache Size: 5
 No Order
 Cycle
 No Randomize
 Wait
SQL> select s3.nextval from employees limit to 13 rows;

                    29
                    30
                    20
                    21
                    22
                    23
                    24
                    20
                    21
                    22
                    23
                    24
                    25
13 rows selected
```

   Note here that the sequence never returns the values 25, 26, 27, 28, 29, 30 after the initial cycle.
2. When a sequence approached the largest positive BIGINT value or the smallest negative BIGINT value, it was possible that an integer overflow could occur and cause the sequence to return incorrect values.
   This example should only generate negative values for the sequence.

```
SQL> show sequence sss;
     SSS
 Sequence Id: 1
 Initial Value: -9223372036854775807
```

```
 Minimum Value: -9223372036854775808
 Maximum Value: -1
 Next Sequence Value: -9223372036854775807
 Increment by: -1
 Cache Size: 20
 No Order
 No Cycle
 No Randomize
 Wait
 Comment:       Return only negative values
SQL> select sss.nextval from rdb$relations limit to 8 rows;

 -9223372036854775807
 -9223372036854775808
  9223372036854775807
  9223372036854775806
  9223372036854775805
  9223372036854775804
  9223372036854775803
  9223372036854775802
8 rows selected
SQL>
```

3. In prior releases, the upper and lower values of the sequence were restricted by subtracting the cache size from the maximum value and adding the cache size to the minimum value in an attempt to avoid overflow errors. This is no longer performed. Therefore, new CREATE CACHE or ALTER CACHE statements that use NOMAXVALUE, MAXVALUE BIGINT, NOMINVALUE, or MINVALUE BIGINT will now default to the largest positive BIGINT value minus one, or the smallest negative BIGINT value plus one. These values (−9223372036854775808, 9223372036854775807) are reserved for use by the sequence generator and may not be specified as a value for the START WITH clause of CREATE SEQUENCE or the RESTART WITH clause of ALTER SEQUENCE. SHOW SEQUENCE may now display a changed value for new sequences.

These problems have been corrected in Oracle Rdb Release 7.2.

3.1.2 Various Errors Using Read Only Areas with Global Buffers

Bug 4630467

Various problems could occur when accessing read only storage areas when global buffers were enabled and multiple processes were accessing the same pages at the same time. For example, bugchecks with exceptions similar to the following could occur:

```
***** Exception at 00EA6948 : RDMS$$EXE_NEXT + 000009D8
%COSI-F-BUGCHECK, internal consistency failure
```

This problem would occur because the buffer was not properly interlocked, allowing multiple users to read data into the buffer at the same time. The contents of the buffer would not be consistent until all I/Os had completed. For very brief instances, the buffer could contain some zeros while it was being filled in by the I/O request. Some users could be accessing the buffer while it contained the transient zeros, leading to various failures or incorrect results.

This problem can be avoided by changing the storage area to be READ WRITE or by disabling global buffers.

This problem has been corrected in Oracle Rdb Release 7.2.

3.1.3 Enhancement for HASH ORDERED Index of BIGINT Column

Bug 3103900

Previously, a HASH ORDERED index of a BIGINT column only considered the least significant 32 bits of the 64 bit value. This lead to unexpected record placement within the storage area. This, in some cases, could lead to severe hash collision problems resulting in excessive I/O and fragmentation issues.

This problem has been corrected in Oracle Rdb Release 7.2. For HASH ORDERED indexes created with Oracle Rdb Release 7.2, an improved algorithm is used when the data type is a BIGINT column. This algorithm considers the full 64−bit value for the hash function. Existing indexes or those indexes not using a BIGINT column retain use of the prior algorithm (utilizing only 32 bits of a 64 bit value).

Customers with databases using HASH ORDERED indexes of a BIGINT column may wish to consider dropping and recreating the index(es) to take advantage of the new algorithm. If the storage map for the table specifies placement via the effected index, then the table should also be reorganized using an unload and reload to ensure correct record placement.

3.1.4 RMU /SHOW LOCKS Limits Relaxed

Bug 3963053

Previously, the "/LOCK=" and "/PROCESS=" qualifiers of the "RMU /SHOW LOCKS" command were limited to 32 specified values.

This problem has been corrected in Oracle Rdb Release 7.2. The "/LOCK=" and "/PROCESS=" qualifiers of the "RMU /SHOW LOCKS" command now accept up to 256 values each.

3.2 SQL Errors Fixed

3.2.1 %SQL−F−UNSDATASS When ALIAS Defined With COMPILETIME PATHNAME

Bug 2315236

Column datatypes for columns defined with one of the ANSI date/time datatypes retrieved from CDD using a COMPILETIME PATHNAME were not correct. Specifically, they always appeared to be of datatype DATE VMS. This caused the generation of a %SQL−F−UNSDATASS during compilation when assignments were made between the affected columns and a properly typed host variable. The problem affected both SQL Module Language and the SQL Precompiler.

Note: You also must run CDD Release 7.2 for this Bug fix to be effective. Otherwise the behavior will be the same as before the fix. That is, SQL$PRE and SQL$MOD will still generate a %SQL−F−UNSDATASS error.

For example, consider the following database definition:

```
VMS> SQL$
SQL> CREATE DATABASE FILENAME TEST_DB PATHNAME TEST_DB;
SQL> CREATE TABLE T_TABLE (T_STAMP TIMESTAMP(2));
SQL> COMMIT;
SQL> EXIT;
```

The following precompiled COBOL program uses a COMPILETIME PATHNAME to retrieve metadata for the T_STAMP column of the T_STAMP table. Note that it also uses a SQL INCLUDE to generate a record named "T_TABLE" from the CDD. This record has the T_STAMP field properly typed. But the column datatype of the T_STAMP column was improperly typed.

```
IDENTIFICATION DIVISION.
PROGRAM−ID.          DATE_TIME_TEST.
AUTHOR.             TEST.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING−STORAGE SECTION.
01  SQLCODE                     PIC S9(9) COMP.
EXEC SQL
  INCLUDE FROM DICTIONARY 'TEST_DB.RDB$RELATIONS.T_TABLE'
END_EXEC.

EXEC SQL DECLARE ALIAS COMPILETIME PATHNAME TEST_DB
                       RUNTIME FILENAME TEST_DB
END_EXEC.

EXEC SQL
   DECLARE TIMESTAMP_TABLE_CURSOR READ ONLY CURSOR FOR
   SELECT * FROM T_TABLE
END_EXEC.

PROCEDURE DIVISION.
00000_ROOT_MODULE.
```

```
EXEC SQL
    OPEN TIMESTAMP_TABLE_CURSOR
END_EXEC.

EXEC SQL
    FETCH TIMESTAMP_TABLE_CURSOR INTO :T_TABLE
END_EXEC.

EXEC SQL
    CLOSE TIMESTAMP_TABLE_CURSOR
END_EXEC.

    STOP RUN.
```

When the program above was compiled, it would generate a *%SQL−F−UNSDATASS* as follows:

```
VMS> SQL$PRE/COBOL TEST_PROGRAM.SCO
    FETCH TIMESTAMP_TABLE_CURSOR INTO :T_TABLE
                                      1
%SQL−F−UNSDATASS, (1) Unsupported date/time assignment from T_STAMP to
T_STAMP IN T_TABLE
```

As a workaround, use COMPILETIME FILENAME.

This problem has been corrected in Oracle Rdb Release 7.2.

3.2.2 SQL Module Language /PROTOTYPES Now Generates INT64 for BIGINT Parameters

Bug 4721570

In prior releases of Oracle Rdb, the SQL Module Language qualifier /PROTOTYPES (or /C_PROTOTYPES) would generate an interface definition that used pointer to long type for the SQL procedure BIGINT parameters. This caused warnings from the C compiler as shown in the following example.

```
MYPROC (sqlstate, &s, &i, &l);
.......................^
%CC−W−PTRMISMATCH, In this statement, the referenced type of the pointer
value "&l" is "__int64", which is not compatible with "long"
at line number 11 in file USER2:[TESTING]SAMPLE.C;1
```

This problem has been corrected in Oracle Rdb Release 7.2. SQL Module Language now generates int64 as the type in the prototype declaration. Applications should include ints.h in their applications and also use int64 when fetching BIGINT data. This is shown in this cut down example.

```
#include <ints.h>
#include <user2:[testing]mod.h>
#include <stdlib.h>

void main ()
{
char sqlstate[5] = "00000";
short s = 0;
int i = 0;
int64 l = 0;
MYPROC (sqlstate, &s, &i, &l);
```

```
exit (0);
}
```

3.2.3 Unexpected Column Ordering Generated by ALTER TABLE ... ALTER COLUMN Statements

Bug 4624762

In prior versions of Oracle Rdb when multiple ALTER COLUMN { BEFORE | AFTER } COLUMN clauses were used in an ALTER TABLE, the resulting column order would be different from that created by multiple ALTER TABLE statements which included just one ALTER COLUMN clause.

The following example shows the problem. Both sets of ALTER TABLE statements should result in the same column ordering.

```
SQL> create table PERSON
cont>     (address char(40)
cont>     ,last_name char(30)
cont>     ,social_security_number integer
cont>     ,first_name char(30)
cont>     );
SQL> commit;
SQL>
SQL> show table (column) PERSON;
Information for table PERSON

Columns for table PERSON:
Column Name                    Data Type        Domain
-----------                    ---------        ------
ADDRESS                        CHAR(40)
LAST_NAME                      CHAR(30)
SOCIAL_SECURITY_NUMBER         INTEGER
FIRST_NAME                     CHAR(30)

SQL>
SQL> alter table PERSON
cont>     alter column first_name
cont>         before column last_name;
SQL> alter table PERSON
cont>     alter column social_security_number
cont>         after column address;
SQL>
SQL> show table (column) PERSON;
Information for table PERSON

Columns for table PERSON:
Column Name                    Data Type        Domain
-----------                    ---------        ------
ADDRESS                        CHAR(40)
SOCIAL_SECURITY_NUMBER         INTEGER
FIRST_NAME                     CHAR(30)
LAST_NAME                      CHAR(30)

SQL>
SQL> rollback;
SQL>
SQL> alter table PERSON
```

```
cont>     alter column first_name
cont>         before column last_name
cont>     alter column social_security_number
cont>         after column address;
SQL>
SQL> show table (column) PERSON;
Information for table PERSON

Columns for table PERSON:
Column Name                    Data Type        Domain
-----------                    ---------        ------
ADDRESS                        CHAR(40)
FIRST_NAME                     CHAR(30)
SOCIAL_SECURITY_NUMBER         INTEGER
LAST_NAME                      CHAR(30)

SQL>
SQL> rollback;
```

This problem has been corrected in Oracle Rdb Release 7.2. Oracle Rdb now preserves the relative ordering of columns when multiple ALTER COLUMN clauses are used.

3.3 RMU Errors Fixed

3.3.1 RMU /UNLOAD Qualifier /REOPEN_COUNT

Bug 4246050

Previously, RMU /UNLOAD would write a single output file for all records in the table/view being unloaded. In some cases, this single output file would become difficult to manipulate.

This problem has been corrected in Oracle Rdb Release 7.2. The "/REOPEN_COUNT=n" qualifier allows one to specify how many records will be written to an output file. The output file will be re−created (ie, a new version of the file created) when the record count reaches the specified value. The "/REOPEN_COUNT=n" qualifier is only valid when used with the "/RECORD_DEFINITION" or "/RMS_RECORD_DEF" qualifiers.

3.3.2 RMU /LOAD /STATISTICS=ON_COMMIT

Previously, the "/STATISTICS=ON_COMMIT" qualifier would have no effect on the RMU /LOAD operations; the qualifier was ignored. The "/STATISTICS=INTERVAL=n" qualifier did function correctly.

This problem has been corrected in Oracle Rdb Release 7.2. When the STATISTICS=(ON_COMMIT) qualifier is specified, Oracle RMU now correctly prints statistics each time a transaction is committed.

Chapter 4
Known Problems and Restrictions

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

4.1 Known Problems and Restrictions in All Interfaces

This section describes known problems and restrictions that affect all interfaces.

4.1.1 External Routines Images Linked with PTHREAD$RTL

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

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

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

- TO_CHAR
- TO_NUMBER
- TO_DATE

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

4.1.2 SQL Procedure External Location Should Be Upper Case

Bug 4722422

When using External Routines, it is important that all declarations for the same shareable image use the exact same strings for the image file specification. Failure to use the same string content may result in multiple copies of the image being activated or failure to correctly call the external routine.

The "ALTER FUNCTION ... LOCATION" command can be used to alter the existing function location string without having to drop and recreate the function.

The following example shows the same string for the EXTERNAL LOCATION specifications:

```
create procedure sys$asctim(
    out :timlen smallint by reference,
    out :timbuf char(23) by descriptor,
    in  :timadr date vms by reference,
```

```
    in  :cvtflag integer by value);
    external location 'SYS$SHARE:SYS$PUBLIC_VECTORS.EXE'
    language general general parameter style;

create procedure sys$gettim(
    in  :timadr date vms by reference);
    external location 'SYS$SHARE:SYS$PUBLIC_VECTORS.EXE'
    language general general parameter style;
```

4.1.3 Using Databases from Releases Earlier than V7.0

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

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

4.1.4 Partitioned Index with Descending Column and Collating Sequence

Bug 2797443

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

The following example can be used to demonstrate the problem.

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

set flags 'strategy';

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

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

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

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

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

Bug 3735144

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

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

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

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

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

In the above example, Rdb can successfully connect to both remote databases using the TCP/IP address "node1.a.b.c." but when multiple databases are attached, Rdb implicitly uses distributed transactions via DECdtm. Since Rdb only passes DECdtm the SCSNODE name retrieved from the RDBSERVERnn at the other end of the connection, DECdtm does not, in general, have the information it needs to resolve the remote

reference. It will only be able to do so if the SCSNODE name and the TCP/IP node name are the same and the local node is on the same subnet (i.e. ".a.b.c" in the example). Otherwise, after the second attach is made, the following error message will be received as soon as a transaction is started:

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

WORKAROUND

There are three potential workarounds:

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

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

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

4.1.6 Some SQL Dialect−required Warnings Not Delivered

Bugs 3651847 and 4532451

The required warnings (information codes) for such things as rows eliminated for nulls (%RDB−I−ELIM_NULL) and string truncation (%RDB−I−TRUN_RTRV) are not being returned for singleton SELECT and singleton UPDATE statements (for example, statements that return a single row using the INTO clause). To demonstrate with a PERSONNEL database, use the following interactive SQL commands:

```
SQL> set dialect 'sql92';
SQL> attach 'filename sql$database';
SQL>
SQL> ! Force a row to contain NULL for SALARY_AMOUNT
SQL> update salary_history
cont> set salary_amount = NULL
cont> where employee_id = '00471'
cont>   and salary_end = date vms'20−Aug−1981';
```

```
1 row updated
SQL>
SQL> declare :avg_sal integer(2);
SQL>
SQL> ! No informational generated (but is expected)
SQL> select avg(salary_amount) into :avg_sal
cont> from salary_history where employee_id = '00471'
cont>  and salary_end >= date vms'1-AUG-1970';
SQL> show sqlca
SQLCA:
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        0
        SQLERRD:        [0]: 0
                        [1]: 0
                        [2]: 1
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       00000
SQL> print :avg_sal;
        AVG_SAL
       60893.86
SQL>
SQL> ! Non singleton query returns correct informational
SQL> select avg(salary_amount)
cont> from salary_history where employee_id = '00471'
cont>  and salary_end >= date vms'1-AUG-1970';

  6.089385714285714E+004
1 row selected
%RDB-I-ELIM_NULL, null value eliminated in set function
SQL> show sqlca
SQLCA:
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        1003
        SQLERRD:        [0]: 0
                        [1]: 0
                        [2]: 1
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       01003
%RDB-I-ELIM_NULL, null value eliminated in set function
SQL>
SQL> rollback;
```

Since there is a row in the SALARY_HISTORY table with a NULL in SALARY_AMOUNT, the set function AVG should report an informational message (and return a special warning level SQLSTATE/SQLCODE value).

```
%RDB-I-ELIM_NULL, null value eliminated in set function
```

4.1.7 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.

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

Bug 2676851

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

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

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

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

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

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

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

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

Galaxy Reboot Required

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

4.1.10 Oracle Rdb and OpenVMS ODS−5 Volumes

The OpenVMS Version 7.2 release 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.

4.1.11 Optimization of Check Constraints

Bug 1448422

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

constraints.

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

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

or:
```

```
SQL> alter table t1
cont>   alter column f1
cont>   check (f1=(select * from t2 where f2=f1))
cont>      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
cont>  after insert on t1
cont>  when (not exists (select * from t2 where f2=f1))
cont>    (error) for each row;
SQL> create trigger t1_update
cont>  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
cont>  before delete on t2
cont>  when (exists (select * from t1 where f1=f2))
cont>    (error) for each row;
SQL> create trigger t2_modify
cont>  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.

4.1.12 Carryover Locks and NOWAIT Transaction Clarification

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

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

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

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

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

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

4.1.14 Both Application and Oracle Rdb Using SYS$HIBER

In application processes that use Oracle Rdb and the $HIBER system service (possibly through RTL routines such as LIB$WAIT), the application must ensure that the event being waited for has actually occurred. Oracle Rdb uses $HIBER/$WAKE sequences for interprocess communications particularly when the ALS (AIJ Log Server) feature is enabled.

The use of the $WAKE system service by Oracle Rdb can 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, consider altering the application to 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 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 BEGIN
         LIB$SIGNAL (STAT)
         END
    ! Hibernate.  When the $HIBER completes, check to make
    ! sure that TIMER_FLAG is set indicating that the wait
    ! has finished.
    WHILE TIMER_FLAG = FALSE
    DO  BEGIN
        SYS$HIBER()
        END
    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 BEGIN
         LIB$SIGNAL (STAT)
         END
    END
```

The LIB$K_NOWAKE flag can be specified when using the OpenVMS LIB$WAIT routine 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.

4.1.15 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.

4.1.16 Excessive Process Page Faults and other Performance Considerations During Oracle Rdb Sorts

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

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

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

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

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

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

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

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

The logical name RDMS$BIND_SORT_WORKFILES specifies how many work files sort is to use if work files are required. The default is 2 and the maximum number is 10. The work files can be individually controlled by the SORTWORKn logical names (where n is from 0 through 9). 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 ten logical names, SORTWORK0 through SORTWORK9.

Normally, SORT places work files in the your SYS$SCRATCH directory. By default, SYS$SCRATCH is the same device and directory as the SYS$LOGIN location. Spreading the I/O load over many disks improves efficiency as well as performance by taking advantage of the system resources and helps prevent disk I/O bottlenecks. Specifying that a your work files reside on separate disks permits overlap of the SORT read/write cycle. You may also encounter cases where insufficient space exists on the SYS$SCRATCH disk device (for example, while Oracle Rdb builds indexes for a very large table). Using the SORTWORK0 through SORTWORK9 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 10 sort work files, it is possible to exceed the capacity of the first SORT file and the sort operation fails never having accessed the remaining 9 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.

4.1.17 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 4−1 Sort Memory Logicals

Logical	Definition
RDMS$BIND_SORT_MEMORY_WS_FACTOR	Specifies a percentage of the process's working set limit to be used when allocating sort memory for the built−in SORT32 package. If not defined, the default value is 75 (representing 75%), the maximum value is 75 (representing 75%), and the minimum value is 2 (representing 2%). Processes with vary large working set limits can sometimes experience significant page faulting and CPU consumption while initializing sort memory. This logical name can restrict the sort work memory to a percentage of the processes maximum working set.
RDMS$BIND_SORT_MEMORY_MAX_BYTES	Specifies an absolute limit to be used when allocating sort memory for the built−in SORT32 package. If not defined, the default value is unlimited (up to 1GB), the maximum value is 2,147,483,647 and the minimum value is 32,768.

4.1.18 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'
cont> 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
cont> 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.

4.2 SQL Known Problems and Restrictions

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

4.2.1 SET FLAGS CRONO_FLAG Removed

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

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

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

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

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

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

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

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

This restriction does not apply to interactive or dynamic SQL.

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

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

```
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.

4.2.4 Restriction for CREATE STORAGE MAP Statement on Table with Data

Oracle Rdb V7.0 added support that allows a storage map to be added to an existing table that contains data. The Oracle Rdb7 Guide to Database Design and Definition describes this feature and lists restrictions.

Oracle Rdb release 7.1 adds the restriction that the storage map cannot include a WITH LIMIT clause for the storage area. The following example shows the resulting error:

```
SQL> create table MAP_TEST1 (a integer, b char(10));
SQL> create index MAP_TEST1_INDEX on MAP_TEST1 (a);
SQL> insert into MAP_TEST1 (a, b) values (3, 'Third');
1 row inserted
SQL> create storage map MAP_TEST1_MAP for MAP_TEST1
cont> store using (a) in RDB$SYSTEM
cont>     with limit of (10);               -- cannot use WITH LIMIT clause
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-RELNOTEMPTY, table "MAP_TEST1" has data in it
-RDMS-E-NOCMPLXMAP, can not use complex map for non-empty table
```

4.2.5 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.

4.2.6 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.

4.3 Oracle RMU Known Problems and Restrictions

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

4.3.1 RMU Convert Fails When Maximum Relation ID is Exceeded

If, when relation IDs are assigned to new system tables during an RMU Convert of an Oracle Rdb V7.0 database 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 V70. 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
```

4.3.2 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.

4.3.3 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.

4.3.4 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.

4.3.5 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.

4.3.6 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.

4.4 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.

4.4.1 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 a maximum of about 600 disk blocks. The actual increase depends mostly on the maximum number of users specified for the database.

4.4.2 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.

4.4.3 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.

4.4.4 Strict Partitioning May Scan Extra Partitions

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

```
ATTACH 'FILENAME MF_PERSONNEL';
CREATE TABLE T1 (ID INTEGER, LAST_NAME CHAR(12), FIRST_NAME CHAR(12));
CREATE STORAGE MAP M FOR T1 PARTITIONING NOT UPDATABLE
 STORE USING (ID)
```

```
 IN EMPIDS_LOW WITH LIMIT OF (200)
 IN EMPIDS_MID WITH LIMIT OF (400)
 OTHERWISE IN EMPIDS_OVER;
INSERT INTO T1 VALUES (150,'Boney','MaryJean');
INSERT INTO T1 VALUES (350,'Morley','Steven');
INSERT INTO T1 VALUES (300,'Martinez','Nancy');
INSERT INTO T1 VALUES (450,'Gentile','Russ');
SELECT * FROM T1 WHERE ID > 400;
Conjunct Get Retrieval sequentially of relation T1
Strict Partitioning: part 2 3
ID LAST_NAME FIRST_NAME
450 Gentile Russ
1 row selected
```

In the previous example, partition 2 does not need to be scanned. This does not affect the correctness of the result. Users can avoid the extra scan by using values other than the boundary values.

4.4.5 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.

4.4.6 Support for Single−File Databases to Be Dropped in a Future Release

Oracle Rdb currently supports both single−file and multifile databases. However, single−file databases will not be supported in a future release of Oracle Rdb. At that time, Oracle Rdb will provide the means to easily convert single−file databases to multifile databases.

Oracle Rdb recommends that users with single−file databases perform the following actions:

- Use the Oracle RMU commands, such as Backup and Restore, to make copies, backup, or move single−file databases. Do not use operating system commands to copy, back up, or move databases.

- Create new databases as multifile databases even though single−file databases are supported.

4.4.7 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.

4.4.8 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.

4.5 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.

4.5.1 SQL Does Not Display Storage Map Definition After Cascading Delete of Storage Area

When you drop a storage area using the CASCADE keyword and that storage area is not the only area to which the storage map refers, the SHOW STORAGE MAP statement no longer shows the placement definition for that storage map.

The following example demonstrates this restriction:

```
SQL> SHOW STORAGE MAP DEGREES_MAP1
     DEGREES_MAP1
 For Table:             DEGREES1
 Compression is:        ENABLED
 Partitioning is:       NOT UPDATABLE
 Store clause:          STORE USING (EMPLOYEE_ID)
            IN DEG_AREA WITH LIMIT OF ('00250')
             OTHERWISE IN DEG_AREA2

SQL> DISCONNECT DEFAULT;
SQL> -- Drop the storage area, using the CASCADE keyword.
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> DROP STORAGE AREA DEG_AREA CASCADE;
SQL> -- Display the storage map definition.
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SHOW STORAGE MAP DEGREES_MAP1
DEGREES_MAP1 For Table: DEGREES1
Compression is: ENABLED
Partitioning is: NOT UPDATABLE
```

The other storage area, DEG_AREA2, still exists, even though the SHOW STORAGE MAP statement does not display it.

A workaround is to use the RMU Extract command with the Items=Storage_Map qualifier to see the mapping.

4.5.2 ARITH_EXCEPT or Incorrect Results Using LIKE IGNORE CASE

When you use LIKE...IGNORE CASE, programs linked under Oracle Rdb V4.2 and V5.1, but run under higher versions of Oracle Rdb, may result in incorrect results or %RDB−E−ARITH_EXCEPT exceptions.

To work around the problem, avoid using IGNORE CASE with LIKE or recompile and relink under a higher version (V6.0 or higher.)

4.5.3 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.

4.5.4 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 4−2 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

4.5.5 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.

4.5.6 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.

4.5.7 AIJSERVER Privileges

For security reasons, the AIJSERVER account ("RDMAIJSERVER") is created with only NETMBX and TMPMBX privileges. These privileges are sufficient to start Hot Standby, in most cases.

However, for production Hot Standby systems, these privileges are not adequate to ensure continued replication in all environments and workload situations. Therefore, Oracle recommends that the DBA provide the following additional privileges for the AIJSERVER account:

- ALTPRI − This privilege allows the AIJSERVER to adjust its own priority to ensure adequate quorum (CPU utilization) to prompt message processing.
- PSWAPM − This privilege allows the AIJSERVER to enable and disable process swapping, also necessary to ensure prompt message processing.
- SETPRV − This privilege allows the AIJSERVER to temporarily set any additional privileges it may need to access the standby database or its server processes.
- SYSPRV − This privilege allows the AIJSERVER to access the standby database rootfile, if necessary.
- WORLD − This privilege allows the AIJSERVER to more accurately detect standby database server process failure and handle network failure more reliably.

| Contents