Oracle Rdb7™

Release Notes

Release 7.0

Oracle Rdb Release Notes

Release 7.0

Contents

Send Us Your Comments

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

You can send comments to us in the following ways:

- Electronic mail — `nedc_doc@us.oracle.com`
- FAX — `603-881-0120 Attn: Oracle Rdb Documentation`
- Postal service

```
Oracle Corporation
Oracle Rdb Documentation
110 Spit Brook Road, ZKO2-1/O19
Nashua, NH  03062-2698
USA
```

If you like, you can use the following questionnaire to give us feedback. (Edit the online release notes file, extract a copy of this questionnaire, and send it to us.)

Name ______________________	Title ______________
Company ____________________	Department ____________
Mailing Address ______________	Telephone Number ________

____

Book Title ___________________	Version Number _________

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

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

Preface

Oracle Rdb software is a general-purpose database management system based on the relational data model.

Purpose of This Manual

This manual contains release notes for Oracle Rdb Version 7.0. The notes describe changed and enhanced features; upgrade and compatiblity information; new and existing software problems and restrictions; and software and documentation corrections.

Note

The release notes are supplied on line in both PostScript and text form.

On systems running OpenVMS VAX or OpenVMS Alpha:

Text	SYS$HELP:RDB070.RELEASE_NOTES
PostScript	SYS$HELP:RDB070_RELEASE_NOTES.PS.

On systems running Digital UNIX:

Text	/usr/lib/dbs/rdb/v70/doc/rdb070.release_notes
PostScript	/usr/lib/dbs/rdb/v70/doc/rdb070_release_notes.ps

Intended Audience

This manual is intended for use by all Oracle Rdb users. Read this manual before you install, upgrade, or use Oracle Rdb Version 7.0.

Structure

This manual consists of four chapters:

Chapter 1	Provides information about changes in installation procedures and licensing and lists new and changed features for this release of Oracle Rdb.
Chapter 2	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb.
Chapter 3	Describes known software errors in versions prior to Oracle Rdb Version 7.0 that are fixed in Version 7.0.
Chapter 4	Provides information not currently available in the Oracle Rdb documentation set.

Conventions

In this manual, Oracle Rdb refers to Oracle Rdb for OpenVMS and Oracle Rdb for Digital UNIX software. Version 7.0 of Oracle Rdb software is often referred to as V7.0.

The SQL interface to Oracle Rdb is referred to as SQL. This interface is the Oracle Rdb implementation of the SQL standard ANSI X3.135-1992, ISO 9075:1992, commonly referred to as the ANSI/ISO SQL standard or SQL92.

Oracle CDD/Repository software is referred to as the dictionary, the data dictionary, or the repository.

OpenVMS means both the OpenVMS Alpha and OpenVMS VAX operating system.

This manual uses icons to identify information that is specific to an operating system or platform. Where material pertains to more than one platform or operating system, combination icons or generic icons are used. For example:

Digital UNIX	This icon denotes the beginning of information specific to the Digital UNIX operating system.
OpenVMS VAX OpenVMS Alpha	This icon combination denotes the beginning of information specific to both the OpenVMS VAX and OpenVMS Alpha operating systems.
♦	The diamond symbol denotes the end of a section of information specific to an operating system or platform.

Although these icons appear in the ASCII-text version of this document, their appearance is different.

In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the Return key at the end of a line of input.

The following conventions are also used in this manual:

Ctrl/x	This symbol indicates that you hold down the Ctrl (control) key while you press another key or mouse button (indicated here by x).
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.
$	The dollar sign represents the DIGITAL Command Language prompt in OpenVMS and the Bourne shell prompt in Digital UNIX.

1
Information About This Release

This chapter highlights changes to the Oracle Rdb kit and the installation procedures and provides information on new and changed features for this release. In addition, it lists the documentation for this release.

1.1 Changes to the Kit and the Installation

The following sections highlight changes to the installation procedure. Please read the *Oracle Rdb7 Installation and Configuration Guide* for installation procedure details.

1.1.1 Names of Oracle Rdb Kits

The names of the Oracle Rdb kits have changed from DECRDBxxxnnn to RDBxxxnnn. The following list shows the name of each kit:

- RDB070: the standard Oracle Rdb for OpenVMS VAX kit
- RDBMV070: the multiversion Oracle Rdb for OpenVMS VAX kit
- RDBA070: the standard Oracle Rdb for OpenVMS Alpha kit
- RDBAMV070: the multiversion Oracle Rdb for OpenVMS Alpha kit

1.1.2 Changes to the Installation Procedure

OpenVMS VAX OpenVMS Alpha

The sale of Rdb from Digital Equipment Corporation to Oracle Corporation has necessitated some modifications to the installation procedure to conform to Oracle licensing policy. Although the menu options have changed, the remainder of the installation procedure has not, so users who have installed prior versions of Oracle Rdb should see no significant differences other than selecting the appropriate Oracle products to install.

As a Digital product, the installation procedure presented the following menu with four installation options:

```
 **********************************************************************

 From the following menu, please select the type of DEC Rdb kit to
 install. Choice CC (COMMON COMPONENTS) is included in all 3 DEC Rdb
 installations and should be selected only when installing a product,
 other than DEC Rdb, that requires the common components.

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

 Please select the type of DEC Rdb kit you wish to install:

        Enter DEV for DEC Rdb DEVELOPMENT (the default)
        Enter INT for DEC Rdb INTERACTIVE
        Enter RTO for DEC Rdb RUNTIME-ONLY
        Enter CC  for COMMON COMPONENTS

* Enter the kit type to install [DEV]:
```

The Oracle Rdb installation procedure now presents you with the following menu:

```
   **********************************************************************

   Please select the Oracle Rdb products you are licensed to install.
   Separate multiple choices with commas (Ex: 1,2,4).

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

    (1)  Oracle Rdb
    (2)  Programmer/2000 (Rdb Compilers)
    (3)  Hot Standby
    (4)  Power Utilities
    (5)  Common Components (For DBI)

* Enter the Oracle Rdb products you are licensed to install [ALL]:
```

Instead of the three DEC Rdb kit types, RUNTIME-ONLY, INTERACTIVE, and DEVELOPMENT, the Oracle Rdb installation menu lists each of its licensed products separately. Select only the products that you are licensed to install. ♦

1.1.3 GBLPAGES System Parameter Value Change

OpenVMS VAX OpenVMS Alpha

The GBLPAGES system parameter value required for installing Oracle Rdb differs on the OpenVMS VAX platform from the OpenVMS Alpha platform.

The number of required pages on OpenVMS VAX is 13,000.

The number of required pages on OpenVMS Alpha is 27,000.

The installation procedure checks for these new values.

Because the *Oracle Rdb7 Installation and Configuration Guide* for V7.0 had been submitted to the printer at the time this information changed, the documented number of required pages in the *Oracle Rdb7 Installation and Configuration Guide* for V7.0 is incorrect. ♦

1.1.4 Oracle Rdb No Longer Checks LMF Information

Oracle Rdb no longer requires a Digital Product Authorization Key (PAK) and no longer performs any License Management Facility (LMF) checking. The installation procedure no longer asks you if you have an authorization key registered and loaded.

1.2 Changes to the Method for Problem Reporting

As of March 1, 1996 all Digital Equipment Corporation support contracts for Oracle Rdb customers expired. If you have a Digital support contract for Oracle Rdb, contact your Oracle support representative.

Because support contracts have changed, Oracle Rdb no longer uses the SPR method for software problem reporting. If you are using a version of Oracle Rdb V6.1 or earlier, you may see messages that advise you to *submit a Software Problem Report (SPR)*. Please disregard these messages.

If an error occurs while you are using Oracle Rdb and you believe that the error is caused by a problem with Oracle Rdb, contact your Oracle support representative for technical assistance.

1.3 Changes in Names of Oracle Rdb Help Files, Command Procedures, Release Notes, and Installation Guides

Because of the purchase of DEC Rdb by Oracle Corporation and the change of the product name to Oracle Rdb, names of many files, such as help files and command procedures, have changed.

Table 1–1 shows the former Digital keywords for accessing help files and the new Oracle keywords.

Table 1–1 Keywords to Access Oracle Rdb Help Files

Type of Help	Digital Keyword	Oracle Keyword
OpenVMS command line help	DECRDB	ORACLE_RDB
Digital UNIX command line help	decrdb	oracle_rdb
Reference page	decrdb	oracle_rdb

The release notes and installation guides have been renamed, as shown in the following table:

From:	To:
DECRDBvvv.RELEASE_NOTES	RDBvvv.RELEASE_NOTES
DECRDBvvv.INSTALL_GUIDE	RDBvvv.INSTALL_GUIDE

The following command procedures have been renamed:

From:	To:
DECRDB$CONVERT_CDD$DATABASE.COM	RDB$CONVERT_CDD$DATABASE.COM
DECRDB$CLUSTER_DEINSTALL.COM	RDB$CLUSTER_DEINSTALL.COM
DECRDB$DEINSTALL_DELETE.COM	RDB$DEINSTALL_DELETE.COM
DECRDB$IVPvv.COM	RDB$IVPvv.COM
DECRDB$SETVER.COM	RDB$SETVER.COM
DECRDB$SHOVER.COM	RDB$SHOVER.COM

1.4 Rdb Web Agent

Oracle Rdb now includes the Rdb Web Agent, which lets you seamlessly invoke SQL stored procedures and produces dynamic HTML pages using SQL. The Rdb Web Agent is implemented using the Common Gateway Interface (CGI), enabling it to function with any Web server that implements a proper CGI. (Some of the servers supported include servers from CERN, Purveyor, Netscape and Oracle.)

You can develop applications using SQL and you can package and treat entire multiscreen applications as single database objects, using the full range of features, including security and recovery, offered by Oracle Rdb.

To use the Rdb Web Agent, you must install a web server and an Oracle SQL/Services client on the same system. You must install Oracle Rdb and the Oracle SQL/Services server on an OpenVMS or Digital UNIX system. (The web server and the Oracle SQL/Services client can be on the same system as Oracle Rdb and the Oracle SQL/Services server, but it is not mandatory.)

Summary information about installing and configuring Rdb Web Agent is provided in the online file rdbweb.install_guide. Comprehensive information about configuring and using the Rdb Web Agent is provided on the software media in HTML format. For more information, see Section 1.15.1.

Oracle Rdb also provides client software for Windows NT Intel, Windows NT Alpha, OpenVMS, and Digital UNIX. For OpenVMS, the Windows NT and Digital UNIX client software, and an accompanying readme.txt file are provided on the Rdb Client kits CD–ROM. The OpenVMS client software is included in the Rdb Web Agent kit.

1.5 Supported Platforms and Network Protocols for PC Clients

Oracle Rdb provides a program group called Oracle Enterprise Manager (OEM) DBAPack that runs as a PC client and includes the following:

- Oracle Rdb Query Performance Tuner graphical user interface

  For more information about the Query Performance Tuner, see Section 1.5.1.

OpenVMS VAX OpenVMS Alpha

- Oracle Rdb Parallel Backup Monitor graphical user interface

  For more information about the Parallel Backup Monitor, see Section 1.5.2. ♦

- Oracle RMUwin graphical user interface

  For more information about Oracle RMUwin, see Section 1.5.3.

- Oracle Rdb Performance Monitor graphical user interface

  For more information about the Performance Monitor, see Section 1.5.3.

- Oracle SQL/Services Manager graphical user interface

  For more information, see the *Oracle SQL/Services Server Configuration Guide*.

- Oracle Rdb OEM Configuration utility

  For more information about the utility, see Section 1.5.4.

For information about installing the DBAPack and the PC client kits, see the readme.txt file on the Rdb Client kit CD–ROM.

The PC clients, except for Rdb Web Agent, are supported on Windows NT Intel 3.5.1, Windows NT Alpha 3.5.1, Windows 95, and Windows 3.1. Rdb Web Agent client software supports only Windows NT Intel 3.5.1 and 4.0 and Windows NT Alpha 3.5.1 and 4.0.

The PC clients on all platforms support both TCP/IP and DECnet communication protocols.

1.5.1 New Query Performance Tuner

For V7.0, Oracle Rdb provides the Query Performance Tuner (QPT), a graphical user interface that enables an Oracle Rdb database administrator to tune individual queries for maximum performance. QPT generates a graphical model of the optimization strategy for an SQL query, and enables the user to modify any aspect of the solution (join order, access paths, join methods, execution strategy). You can save the strategy in the database. Oracle Rdb applies it on subsequent compilations and executions of the query.

The Query Performance Tuner requires Oracle Rdb V6.0 or later, although query outline generation and minimization are supported only for Oracle Rdb V7.0 and later versions.

For information about installing the DBAPack, including QPT, see the readme.txt file on the Rdb Client kits CD–ROM.

After you install the DBAPack, you invoke QPT by double clicking on the icon.

For information about supported platforms and network support for each platform, see Section 1.5.

1.5.2 New Parallel Backup Monitor

OpenVMS VAX OpenVMS Alpha

Oracle Rdb provides the Parallel Backup Monitor, which lets you monitor the progress of parallel backup operations, collect information for future operations, and determine bottlenecks.

For information about installing the DBAPack, including the Parallel Backup Monitor, see the readme.txt file on the Rdb Client kits CD–ROM.

After you install the DBAPack, you invoke the Parallel Backup Monitor by double clicking on the icon.

For information about supported platforms and network support for each platform, see Section 1.5. ♦

1.5.3 Oracle RMUwin, Performance Monitor Available on Windows Only

Oracle Rdb provides support for the new client/server versions of Oracle RMUwin and the Performance Monitor on Windows NT Intel 3.5.1, Windows NT Alpha 3.5.1, Windows 95, and Windows 3.1.

Note

Oracle RMUwin and the Performance Monitor for Oracle Rdb V7.0 do not support the DECwindows Motif software interface. These tools will continue to support the Motif software on systems running Oracle Rdb V6.1. See Section 2.3.5 for more information about Motif support.

For information about network support for each platform, see Section 1.5.

For information about installing the DBAPack, which includes Oracle RMUwin and the Performance Monitor, see the readme.txt file on the Rdb Client kits CD–ROM.

After you install the Oracle Enterprise Manager DBAPack, you invoke RMUwin or the Performance Monitor by double clicking on the icons.

1.5.4 Launching Applications from Oracle Enterprise Manager

You can now launch DBAPack applications from the Oracle Enterprise Manager (OEM). The DBAPack program group now includes the OEM Configuration utility, which lets you launch DBAPack applications and Oracle Trace and Oracle Expert for Rdb from Oracle Enterprise Manager.

For more information, see the online help for the Oracle Rdb OEM configuration Utility.

For information about installing the DBAPack, which includes the OEM configuration utility, see the readme.txt file on the Rdb Client kits CD–ROM. For information about supported platforms and network support for each platform, see Section 1.5.

1.6 New Hot Standby Database Option

OpenVMS VAX OpenVMS Alpha

Oracle Rdb V7.0 introduces the Oracle Hot Standby option which is a discrete, separately-purchasable product that provides high-performance database replication. The Hot Standby option prevents your Oracle Rdb database or Oracle CODASYL DBMS database from becoming a single point of failure by physically duplicating the database at a geographically remote standby site. In the event of a node or cluster failure, the replicated hot standby database automatically becomes the master database and takes over application processing.

Note

Neither the master database nor the standby database is affected by a failure of the other; a system failure of the master database is isolated from the hot standby database and vice versa.

The Hot Standby option automatically performs coordinated database synchronization with high performance and minimal impact on system resources. You need to intervene manually only to start the replication services and to fail over applications to the standby database if a failure occurs. The Hot Standby option does not require specific hardware to operate and you do not need to make any changes to application coding.

For Oracle Rdb, you start the replication operation by entering a Replicate command using Oracle RMU.

You can implement this feature on multiple operating system platforms. For Oracle Rdb, the master and standby databases can be implemented on systems running OpenVMS VAX, OpenVMS Alpha, or any combination of these operating systems.

Refer to the *Oracle Rdb7 and Oracle CODASYL DBMS: Guide to Hot Standby Databases* for complete information about the Hot Standby option.

Note

Online help for the Hot Standby option is available in HTML format only. You can find the Replicate command reference information in SYS$HELP:HOT_STANDBY.HTML

♦

1.7 SGA API Available for Oracle Rdb

OpenVMS VAX OpenVMS Alpha Oracle Rdb provides a system global area (SGA) application programming interface (API) which gives internal applications and third-party developers a supported way to retrieve database performance statistics.

Oracle Rdb maintains an extensive set of online performance statistics that provide valuable dynamic information regarding the status of an active database. The SGA API provides a way to access these statistics quickly.

The Oracle Rdb SGA API is provided as a saveset on the Oracle Rdb kit. The saveset is named RDBSGA_xx_ptt.BCK, where:

xx	Oracle Rdb version. For example, 70 for Version 7.0
p	Platform–V for VAX, or A for Alpha
tt	Kit type–MV for multiversion, or S for standard

To use the API, restore the saveset to your working directory with the following command:

```
$ BACKUP/LOG [kit_directory]RDBSGA_xx_ptt.BCK/SAVE []*.*;0
```

When you have restored the saveset, read the $$$RDBSGA_API.README file for further instructions. ♦

1.8 New Features Affecting All Interfaces

This section summarizes new and changed features that affect all interfaces to Oracle Rdb. These features include:

- New on-disk structure for B-tree (sorted) indexes

  In V7.0, you can specify that Oracle Rdb use a new ranked B-tree structure. The new structure allows better optimization of queries, particularly queries involving range retrievals. Oracle Rdb makes better estimates of cardinality, reducing disk I/O and lock contention. For more information, see the *Oracle Rdb7 Guide to Database Design and Definition* and *Oracle Rdb7 SQL Reference Manual*.

- Duplicates compression

  If a ranked sorted index allows duplicates, you can store many more records in a small space by using duplicates compression. When you do, Oracle Rdb uses byte-aligned bitmap compression to represent the dbkeys for the duplicate entries, instead of chaining the duplicate entries with uncompressed dbkeys. In addition to the savings in storage space, you minimize I/O, increasing performance. For more information, see the *Oracle Rdb7 Guide to Database Design and Definition*.

- Improved duplicate detection

  If a unique index is defined on a table and you try to insert a duplicate record, Oracle Rdb intercepts the action more quickly. Because Oracle Rdb rolls back precreated duplicate nodes, the database will not have excess locked free space.

- Specifying an alternate TCP/IP service

By specifying an alternate TCP/IP service, you can access different versions of OpenVMS databases through TCP/IP from an OpenVMS or Digital UNIX client. You can also use the alternate service for any other special access requirements that are not met by the default TCP/IP service. For more information, see Section 1.8.1.

- CREATE INDEX optimization for empty tables

  Oracle Rdb avoids scanning all areas if a table is empty and if the table was created in the current transaction. Oracle Rdb maintains an internal list of newly created tables to determine whether or not the table contains data. For more information, see Section 1.8.2.

- Support for quadword TSNs and CSNs

  Prior to this release, Oracle Rdb represented a transaction sequence number (TSN) or a commit sequence number (CSN) as a longword value. The maximum value for a TSN or CSN was 4,294,967,295. Beginning with this release, Oracle Rdb represents both TSNs and CSNs as quadword values, in the following decimal format:

  ```
  high longword : low longword
  ```

  The high longword can hold a maximum user value of 32768 (2^{15}) and the low longword can hold a maximum user value of 4,294,967,295 (2^{32}). A portion of the high-longword is used by Oracle Rdb for overhead.

  When you specify a TSN or CSN, you can omit the high longword and the colon if the TSN or CSN fits in the low longword. All of the following are valid TSN or CSN input values:

  ```
  0:444
  444
  2:555
  ```

  The following shows a series of TSN and CSN values, in ascending order:

  ```
  0:1
  0:4294967295
  1:0
  1:4294967295
  2:0
  ```

  The TSN values are *not* represented with this format when you use the RdbAlter commands. See the documentation for the RdbAlter Deposit and RdbAlter Display commands in the *Oracle RMU Reference Manual* for details.

Digital UNIX

- Support for distributed transactions that conform to the XA specification

  Oracle Rdb provides support for transactions that conform to the XA specification of the X/Open standard. For more information, see Section 1.8.5. ♦

- New logical name, RDMS$BIND_PRESTART_TXN, and configuration parameter, RDB_BIND_PRESTART_TXN

  This logical name and configuration parameter allow you to establish the default setting for pre-started transactions outside an application. For more information, see the *Oracle Rdb7 Guide to Database Performance and Tuning*.

- New logical name, RDM$BIND_RUJ_ALLOC_BLKCNT, and configuration parameter, RDB_BIND_RUJ_ALLOC_BLKCNT

This logical name and configuration parameter define the .ruj file initial allocation size, expressed in blocks. The minimum value is 1, the maximum value is 2 billion, and the default value is 127 blocks. For more information, see the *Oracle Rdb7 Guide to Database Performance and Tuning*.

- Optimizer statistics

  Oracle Rdb collects new statistics that help the optimizer to significantly reduce errors in cost and cardinality estimation, generally improving the query optimization process and increasing the probability of finding the optimal solution for each query.

  The statistics also reduce the volatility (degree of change) in query strategy generation exhibited by the optimizer when you upgrade to a higher Oracle Rdb version. The use of new statistics by the optimizer translates into overall improvement in performance across the query workload.

  The new statistics capture the distribution of data values in interesting column groups of various base tables and the clustering of data in the sorted and hashed indexes, as well as in the table areas. The interesting column groups, called **workload column groups**, are identified from various queries of a workload.

  The optimizer statistics, including the new statistics, can be classified into three categories: basic, workload, and storage statistics. For more information, see the *Oracle Rdb7 Guide to Database Performance and Tuning*.

- Row-level memory cache

  The row-level memory cache feature allows frequently referenced rows to remain in memory even when the associated page has been flushed back to disk. This saves memory usage because only the more recently referenced rows are cached versus caching the entire buffer.

  You can specify whether or not large memory is used to manage the row cache. **Large memory** allows Oracle Rdb to use as much physical memory as available and to dynamically map it to the virtual address space of database users. It provides access to a large amount of physical memory through small virtual address windows.

  For more information, see the *Oracle Rdb7 Guide to Database Performance and Tuning* and the *Oracle Rdb7 SQL Reference Manual*.

OpenVMS Alpha

- A new type of global section, system space buffers (SSB)

  The system space global section is located in the OpenVMS Alpha system space, which means that a system space global section is fully resident, or pinned, in memory and does not affect the quotas of the working set of the process. As a result, a process referencing a system space global section can have up to 256 Mb of resident working set space.

  For more information, see the *Oracle Rdb7 Guide to Database Performance and Tuning*. ♦

- Index-only retrieval for hashed indexes

  Oracle Rdb V7.0 uses an index-only strategy to retrieve the data from the Ikey segments of a hash index, where all the necessary data records are defined. Thus, it eliminates the extra I/O to fetch the data record using the database key (dbkey) from the index node, and significantly improves the performance.

The following example shows the index-only retrieval strategy using the hash index EMPLOYEES_HASH, over sorted index EMP_EMPLOYEE_ID. The cost of the chosen solution is less than the sorted counterpart.

```
SELECT R.EMPLOYEE_ID FROM  EMPLOYEES R
  WHERE R.EMPLOYEE_ID = '00166' OR  R.EMPLOYEE_ID = '00177';
Solutions tried 4
Solutions blocks created 2
Created solutions pruned 1
Cost of the chosen solution      0.0000000E+00
Cardinality of chosen solution   1.9900000E+00
Conjunct        Index only retrieval of relation EMPLOYEES
  Index name  EMPLOYEES_HASH [1:1...]2
 EMPLOYEE_ID
 00166
 00177
2 rows selected
```

- Eliminating redundant sort in singleton select

  Oracle Rdb V7.0 eliminates the redundant sort that the optimizer applies when the select expression specifies ORDER BY to order the output of two streams where one stream returns one single row (singleton) and the other stream uses an ordered B-tree index retrieval. See Section 1.8.3 for more information.

- Eliminating extra sorts

  The optimizer now uses a minimal number of sorts when processing queries with GROUP BY, ORDER BY, and DISTINCT clauses. For example:

```
SQL> SELECT SUM(MAXIMUM_SALARY) FROM
cont>    (SELECT DISTINCT MAXIMUM_SALARY FROM JOBS) AS MS
cont>    GROUP BY MAXIMUM_SALARY;
Aggregate
Merge of 1 entries
  Merge block entry 1
  Reduce  Sort    Get     Retrieval sequentially of relation JOBS
```

- 64-Bit cardinality

  Oracle Rdb V7.0 now handles tables and indexes with approximately 4 billion rows and beyond. This new feature is made possible by increasing the cardinality size to 64-bit from 32-bit.

- Changes in cardinality update algorithm

  In V7.0, Oracle Rdb has revised its update algorithms to reduce the I/O performed to update cardinalities when cardinality collection is enabled.

  See Section 1.8.4 for more information.

- Zigzag match join enhancements for better performance

  Oracle Rdb now uses zigzag skip on the outer loop as well as the inner loop. In addition, it uses an intermediate result table for the equal-key group join instead of index rescan. The following example demonstrates this capability and includes output from the RDMS$DEBUG_FLAGS logical name:

```
SQL> SELECT * FROM EMPLOYEES E, SALARY_HISTORY S
cont>    WHERE E.EMPLOYEE_ID = S.EMPLOYEE_ID;
Conjunct
Match
  Outer loop      (zig-zag)
    Get     Retrieval by index of relation EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Inner loop      (zig-zag)
    Get     Retrieval by index of relation SALARY_HISTORY
      Index name  SH_EMPLOYEE_ID [0:0]
```

1.8.1 SQL_ALTERNATE_SERVICE_NAME Configuration Parameter

In previous versions, users of the TCP/IP network protocol were restricted to using the default rdbserver service for all remote database access.

Oracle Rdb V7.0 includes a new configuration parameter, SQL_ALTERNATE_ SERVICE_NAME, that lets you specify an alternate TCP/IP service. This is especially useful for accessing different versions of OpenVMS databases through TCP/IP from an OpenVMS or Digital UNIX client. You can also use it for any other special access requirements that are not met by the default rdbserver TCP/IP service.

The following example shows the line you add to your client configuration file to use the new service myservice, which you have defined for TCP/IP:

```
SQL_ALTERNATE_SERVICE_NAME myservice
```

OpenVMS VAX OpenVMS Alpha

On OpenVMS systems, you must use UCX to create your new service. Refer to the *Oracle Rdb7 Installation and Configuration Guide* for more information on setting up services in UCX. To set up a service for accessing different versions of Oracle Rdb databases (V6.1 in this example), you must make sure that the LOGIN.COM of the user name defined for the service contains the following lines:

```
$ DEFINE RDBSERVER SYS$SYSTEM:RDBSERVER61.EXE
$ DEFINE RDMS$VERSION_VARIANT 61
```
♦

Digital UNIX

On Digital UNIX systems, if you use an alternate service to access an OpenVMS database, you must define the service on your local node and on the remote OpenVMS node. On your local node, add the service to the /etc/services file. On the OpenVMS node, use the UCX utility to define the service.

On Digital UNIX systems, if you use an alternate service to access a Digital UNIX database, you must define the service in the /etc/services file on the local and remote nodes. You must also add the service (myservice, in this example) to the remote system's /etc/inetd.conf file, as shown in the following example:

```
myservice  stream  tcp  nowait  dbsmgr \
  /usr/lib/dbs/sql/v70/lib/rdbserver v70/rdbserver
```

In addition, on the remote node you must send a hangup to the inet daemon to reinitialize it so that it will see your addition. Use the following commands to do this:

```
ps ax | grep inetd
  (results will show the PID running inetd - for example, 515)

kill -HUP 515
```

Please note that the user name for your alternate service *must* be dbsmgr.

If you need to specify an alternate server configuration file for your service, rdbserver requires an optional command line. To use /usr/jones/sql.conf instead of the default sql.conf for dbsmgr, you specify it as shown in the following example:

```
myservice  stream  tcp  nowait  dbsmgr \
  /usr/lib/dbs/sql/v70/lib/rdbserver \
  v70/rdbserver -s /usr/jones/sql.conf
```

♦

1.8.2 CREATE INDEX Optimization for Empty Tables

In previous versions, a CREATE INDEX statement checked whether or not a table was empty by fetching the first row. If the table was empty, Oracle Rdb could avoid collection and sorting of the data and creation of the index. This optimization works very well in uniform storage areas where the SPAM pages allow fast access to the first row.

However, in mixed-format storage areas, each page must be read and checked for an occurrence of a row for the table. In particular, when the table is partitioned across many areas, CREATE INDEX could execute many I/O operations to determine whether or not a table was empty.

V7.0 includes an optimization within CREATE INDEX to avoid this area scan for all areas. If the table was created in the current transaction, sufficient internal information exists for Oracle Rdb to know if a table does not contain data. Oracle Rdb maintains an internal list of newly created tables to support this optimization.

If you attach to the database using RESTRICTED ACCESS, Oracle Rdb carries this optimization through until you disconnect from the database. Restricted access is necessary so that Oracle Rdb can guarantee that no other process has updated the table after it was created during this session. Because the IMPORT statement, by default, attaches to the new database using RESTRICTED ACCESS, this optimization helps improve the import performance of large databases. In particular, fewer I/O operations are now needed to import a table that is placed using a hashed index if that table is mapped to mixed-format areas.

In some cases where applications create and drop many tables, the maintenance of the internal list of new tables may not be desirable. In those cases, you can define the logical name RDMS$USE_OLD_COUNT_RELATION or the configuration parameter RDB_USE_OLD_COUNT_RELATION to disable this optimization. Only the existence of the logical name or configuration parameter is required; it can be defined as any value.

1.8.3 Eliminating Redundant Sort in Singleton Select

Oracle Rdb V7.0 eliminates the redundant sort that the optimizer applies when the select expression specifies the ORDER BY clause to order the output of two streams where one stream returns one single row (singleton) and the other stream uses an ordered B-tree index retrieval.

Singleton means a query stream that outputs a single row, such as scalar aggregate, direct (unique) key lookup, and dbkey retrieval. The following examples show different types of query streams.

The following example shows a query with scalar aggregate stream:

```
SQL> SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME,
cont>    (SELECT AVG(SALARY_AMOUNT) FROM SALARY_HISTORY)
cont>        FROM EMPLOYEES
cont>        ORDER BY EMPLOYEE_ID LIMIT TO 3 ROW;
Cross block of 2 entries
  Cross block entry 1
    Aggregate       Get     Retrieval sequentially of relation  SALARY_HISTORY
  Cross block entry 2
    Firstn  Get     Retrieval by index of relation EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
 EMPLOYEE_ID   FIRST_NAME   LAST_NAME
 00164         Alvin        Toliver           2.652896707818930E+004
 00165         Terry        Smith             2.652896707818930E+004
 00166         Rick         Dietrich          2.652896707818930E+004
```

The following example shows a query with a direct lookup stream:

```
SQL> CREATE UNIQUE IND EMP_LAST_FIRST ON EMPLOYEES (LAST_NAME, FIRST_NAME);
SQL> DROP INDEX EMP_EMPLOYEE_ID;

SQL> SELECT LAST_NAME, FIRST_NAME, J.JOB_CODE FROM EMPLOYEES E, JOB_HISTORY J
cont>  WHERE E.EMPLOYEE_ID = J.EMPLOYEE_ID AND
cont>     E.LAST_NAME = 'Ziemke' and E.FIRST_NAME = 'Al'
cont>  ORDER BY E.EMPLOYEE_ID;

Cross block of 2 entries
  Cross block entry 1
    Get     Retrieval by index of relation EMPLOYEES
      Index name  EMP_LAST_FIRST [2:2]       Direct lookup
  Cross block entry 2
    Leaf#01 BgrOnly JOB_HISTORY Card=274
      BgrNdx1 JH_EMPLOYEE_ID [1:1] Fan=17
```

The following example shows a query with multiple CROSS streams:

```
SELECT E.EMPLOYEE_ID,
         (SELECT COUNT(*) FROM JOB_HISTORY),
         (SELECT COUNT(*) FROM JOB_HISTORY JH
            WHERE JH.EMPLOYEE_ID=E.EMPLOYEE_ID),
         (SELECT COUNT(*) FROM DEGREES),
         (SELECT COUNT(*) FROM DEGREES D
            WHERE D.EMPLOYEE_ID = E.EMPLOYEE_ID)
    FROM EMPLOYEES E LIMIT TO 5 ROW;

Match
  Outer loop
    Cross block of 4 entries
      Cross block entry 1
        Aggregate       Index only retrieval of relation DEGREES
          Index name  DEG_COLLEGE_CODE [0:0]
      Cross block entry 2
        Aggregate       Index only retrieval of relation JOB_HISTORY
          Index name  JH_EMPLOYEE_ID [0:0]
      Cross block entry 3
        Firstn  Index only retrieval of relation EMPLOYEES
          Index name  EMP_EMPLOYEE_ID [0:0]
      Cross block entry 4
        Aggregate       Index only retrieval of relation JOB_HISTORY
          Index name  JH_EMPLOYEE_ID [1:1]
  Inner loop      (zig-zag)
    Aggregate       Index only retrieval of relation DEGREES
      Index name  DEG_EMP_ID [0:0]
```

The following example shows a query with a dbkey retrieval stream:

```
CREATE VIEW V1 AS SELECT EMPLOYEE_ID,LAST_NAME FROM EMPLOYEES;
DECLARE :DBK BIGINT;
SELECT DBKEY INTO :DBK FROM V1 WHERE EMPLOYEE_ID ='00166';
SELECT LAST_NAME, FIRST_NAME, J.JOB_CODE FROM EMPLOYEES E, JOB_HISTORY J
  WHERE E.EMPLOYEE_ID = J.EMPLOYEE_ID AND E.DBKEY = :DBK
          ORDER BY E.EMPLOYEE_ID;
Cross block of 2 entries
   Cross block entry 1
     Conjunct       Firstn  Get     Retrieval by DBK of relation EMPLOYEES
   Cross block entry 2
     Leaf#01 FFirst JOB_HISTORY Card=274
       BgrNdx1 JH_EMPLOYEE_ID [1:1] Fan=17
```

1.8.4 Changes in Cardinality Update Algorithm

As rows are deleted or inserted into a table, or as key values change for indexes, Oracle Rdb collects cardinality statistics for use by the optimizer. At each commit, a decision is made to write these collected statistics to the tables RDB$RELATIONS, RDB$INDICES and RDB$INDEX_SEGMENTS or to cache them until a later commit. Ultimately, the statistics are flushed to disk when the database is disconnected.

In previous versions, the statistics were flushed to disk whenever a cardinality difference (inserts - deletes) was within log base 2 of the table's current cardinality. This threshold increases relatively slowly with respect to the linear increase in table cardinality. Even for very large tables, a difference in excess of only 64 rows caused the cardinality data to be updated on disk during the next commit.

For high update environments, this extra I/O to the RDB$SYSTEM storage area is often not desired. Options exist to change the RDB$SYSTEM storage area to read-only or to disable cardinality collection using the CARDINALITY COLLECTION IS DISABLED clause of the SQL ALTER DATABASE statement. You can use the RMU Collect Optimizer_Statistics command to update these statistics at a more convenient time.

In V7.0, Oracle Rdb has revised its update algorithms to reduce the I/O performed to update cardinalities when cardinality collection is enabled:

- For low cardinality tables (1024 rows), the algorithm remains the same.
- For high cardinality tables, Oracle Rdb does not flush the cardinality update to disk if it less than 1% of the table cardinality. As the table grows in size, less I/O is required to maintain the cardinalities. This 1% difference between actual and recorded cardinality can be tolerated by the optimizer.
- For very high cardinality tables, the cardinality difference cannot exceed the 1% threshold, even after a large number of commits. By default, a table or index whose cardinality changes have not been flushed in the last 100 commit statements is flushed to disk.

  You can define the logical name RDMS$BIND_CARD_UPDATE_QUOTA or the configuration parameter RDB_BIND_CARD_UPDATE_QUOTA to a positive number which represents the number of COMMIT statements to be executed before cardinalities are flushed to disk.

  If the number is low, the update frequency to RDB$RELATIONS, RDB$INDICES and RDB$INDEX_SEGMENTS is higher. If the number is large, less I/O is used to keep track of the cardinalities with a trade-off against the relative accuracy of the stored cardinality.

1.8.5 Support for XA Transactions

Digital UNIX

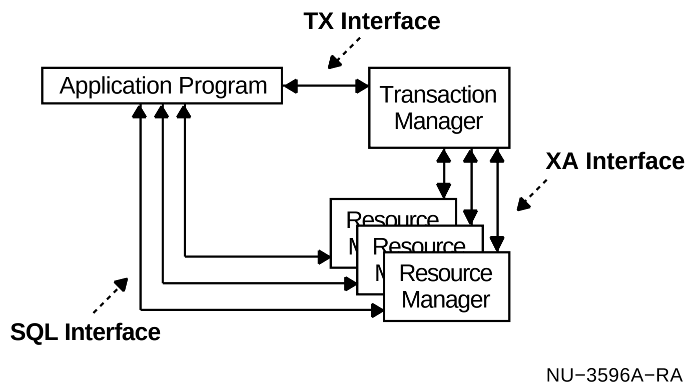
Oracle Rdb provides support for transactions that conform to the XA specification of the X/Open standard.

X/Open is an independent organization that publishes standards for the information systems industry.

The **XA Specification** describes facilities by which commercial applications can achieve distributed transaction processing using the two-phase commit protocol. The specification describes the interface between a **transaction manager** and a **resource manager**. The specification was last released in 1991.

Figure 1–1 shows the parts of an XA transaction.

Figure 1–1 Parts of an XA transaction



The XA Specification does *not* describe the interface between the application program and the database resource manager— it varies from one database vendor to the other. For Oracle Rdb, that interface is SQL.

The interface between the application program and the transaction manager (sometimes called the TX interface) is not yet standardized and varies from one transaction manager vendor to another.

Prior to V7.0, Oracle Rdb provided support for distributed transactions only on OpenVMS using DECdtm software. In V7.0, Oracle Rdb supports the Encina transaction manager from Transarc. The *Oracle Rdb7 Guide to Distributed Transactions* describes distributed transactions and provides some conceptual information that is pertinent to both DECdtm and Encina transaction managers.

Before using XA transactions, you should become familiar with the XA specification and with the Encina transaction manager.

1.8.5.1 Using Oracle Rdb with XA Transaction Managers

Beginning with V7.0, Oracle Rdb supports the Encina transaction manager from Transarc. Using Encina, Oracle Rdb databases on Digital UNIX and OpenVMS can participate in distributed transactions. You can use Encina with Oracle Rdb for the following distributed transactions:

- A transaction that attaches to more than one Oracle Rdb database on Digital UNIX or OpenVMS
- A transaction that attaches more than once to an Oracle Rdb database on Digital UNIX or OpenVMS

- A transaction that attaches to one or more Oracle Rdb for Digital UNIX databases and one or more Oracle Rdb for OpenVMS databases

  Because Encina is not implemented on OpenVMS, your application must run on Digital UNIX. However, the application can attach to OpenVMS databases and involve them in the distributed transaction.

Note the following points about using Oracle Rdb with the Encina transaction manager:

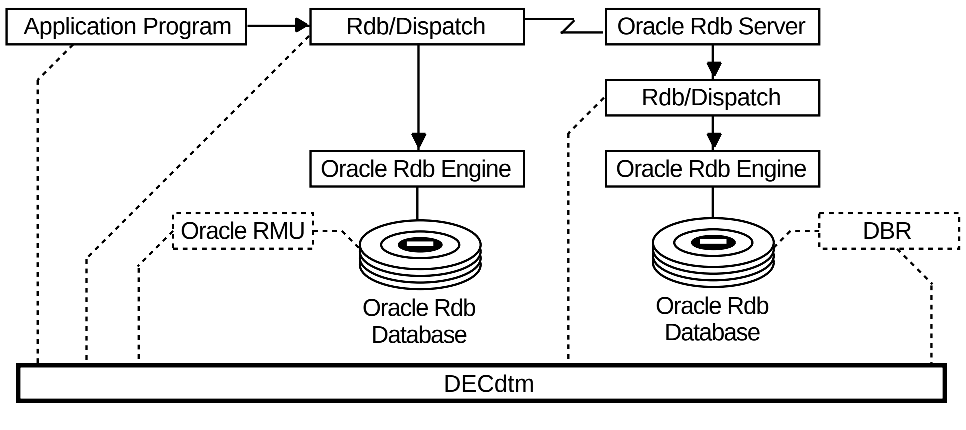
- Only databases that are created in V7.0 or higher, or have been converted to V7.0 or higher, can participate in XA transactions.
- To start a distributed transaction, you must have the DISTRIBTRAN database privilege for all databases involved in the transaction.
- Oracle Rdb supports only explicit distributed transactions with Encina. This means that your application must explicitly call the Encina routines to start and end the transactions. In addition, it means that you cannot use interactive SQL.

1.8.5.2 Relationship of Oracle Rdb Components to DECdtm

Prior to V7.0, Oracle Rdb provided support for distributed transactions only on OpenVMS using DECdtm software.

Figure 1–2 shows how Oracle Rdb communicates with DECdtm, which acts as the transaction manager.

Figure 1–2 Oracle Rdb with DECdtm



On OpenVMS, the application program communicates with Rdb/Dispatch and (in the case of explicit distributed transactions) directly with DECdtm. As the transaction progresses, Rdb/Dispatch communicates with DECdtm both on the local node and in the remote server process. DECdtm is aware of the location of each branch of the transaction.

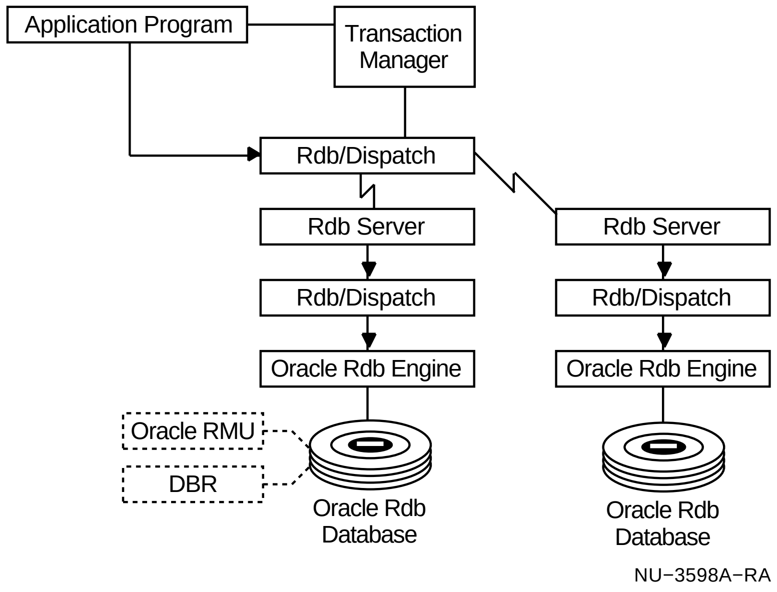
During recovery, the database recovery (DBR) process can call on DECdtm to tell it whether a transaction branch should be committed or rolled back. The Oracle RMU utility can call on DECdtm to find the status of a transaction.

DECdtm stores information about the application's transaction, along with all the other applications running on the network, in a distributed database.

1.8.5.3 Relationship of Oracle Rdb Components to an XA Transaction Manager

The relationship of Oracle Rdb components to an XA transaction manager is different from those components' relationship to DECdtm. Figure 1–3 shows how Oracle Rdb communicates with XA transaction managers.

Figure 1–3 Oracle Rdb with XA



The application program communicates with Rdb/Dispatch and with the transaction manager. The transaction manager has no information on the location of each branch of the transaction, but only communicates with Oracle Rdb on the node on which the application is running.

During recovery, the database recovery (DBR) process must wait until the transaction manager tells it what to do, by way of another Oracle Rdb server process. The DBR process cannot initiate a conversation with the transaction manager to find out what is occurring.

The Oracle RMU utility cannot find out anything about transactions beyond what is stored in the individual databases.

The transaction manager is closely bound to the application program and stores information only about that application's transactions. Other applications use separate instances of the transaction manager.

1.8.5.4 Using SQL with an XA Transaction Manager

To use the XA transaction manager in your applications, use SQL module language programs or precompiled SQL programs.

Because the XA and SQL interfaces to the Rdb/Dispatch component of Oracle Rdb evolved independently, their concepts are different. Rdb/Dispatch must match the concepts coming from the transaction manager (through the XA entry points) and the application program (through internal calls from SQL). The two main areas of concern are *database* references (attaches) and *transaction* references.

Database References

When attaching through SQL, a program presents a *database name* (and optional security information). Oracle Rdb returns a *database handle* for use in subsequent calls to refer to the database. SQL keeps this handle in its internal connection context.

At the corresponding XA entry point (xa_open), a transaction manager presents an *open_info* string and a *resource manager ID* (rmid) number. In subsequent calls, the rmid is used to refer to the database. The name of the database is derived from the contents of the *open_info* string.

Each database vendor must specify the format of an *open_info* string; for Oracle Rdb, the format is:

```
Oracle_Rdb_XA+DB=<dbname>+User=<username>+Pwd=<password>
```

The following shows an example of a valid open_info string:

```
Oracle_Rdb_XA+DB=personnel+User=jones+Pwd=mypasswd
```

The open_info string must always begin with the string `Oracle_Rdb_XA`. The following describes the arguments of the open_info string:

- `+DB=<dbname>`

  A required argument that identifies the database name, including node name, optional access information such as user name and password, and the file specification.

  The value must be exactly the same string that is supplied in the SQL statement that specifies the databases to which the application will attach. In this way, Rdb/Dispatch can match the *database handle* used by the application program with the *rmid* used by the transaction manager.

  The argument name, DB, is case sensitive. The value of the <dbname> argument is case sensitive for Digital UNIX systems, but not case sensitive for OpenVMS systems.

- `+User=<username>`

  An optional argument that identifies the user account and is used for user authentication. The argument name, User, is case sensitive. The value of the <username> argument is case sensitive for Digital UNIX systems, but not case sensitive for OpenVMS systems.

- `+Pwd=<password>`

  An optional argument that identifies the user's password and is used for user authentication. The argument name, Pwd, is case sensitive. The value of the <password> argument is case sensitive for Digital UNIX systems, but not case sensitive for OpenVMS systems.

Oracle Rdb uses the <username> and <password> arguments to attach to the database during recovery, which is initiated by the transaction manager.

Oracle Rdb requires that the xa_open call occur *before* the corresponding SQL statement that attaches to the database.

Transaction References

At the SET TRANSACTION entry point, a program presents one or more *database handles* (along with other information, such as locking modes). Oracle Rdb returns a *transaction handle*. SQL adds this transaction handle to the information in its connection context.

At the corresponding XA xa_start entry point, a transaction manager presents one *rmid* and one *transaction identifier* (XID). If more than one database is involved, more xa_start calls are made.

Rdb/Dispatch matches these calls by comparing the *database handles* supplied by SQL with the *rmids* supplied by the XA interface. Clearly, this only works if the *rmid* values refer to exactly the same database attachment as the *database handle* values, and this only works if the database references were correctly matched.

Ordering Database and Transaction References

To allow Rdb/Dispatch to match XA references to SQL references, you must perform certain steps in a specified order:

1. Initialize the transaction manager and use the open_info string to identify the databases *before* you explicitly attach to the database using an SQL CONNECT or ATTACH statement.

   Oracle Rdb recommends that you use the CONNECT statement, because it provides greater control and less overhead than the ATTACH statement.

   When you use the CONNECT statement, remember to use the `-s '-conn'` switch on the SQL precompiler command line or the `-conn` switch on the SQL module processor command line.

2. Direct the transaction manager to start a transaction *before* you use an SQL SET TRANSACTION statement.

   Use the SET TRANSACTION statement rather than the DECLARE TRANSACTION statement to explicitly start the transaction.

Avoid implicit attaches and transactions in SQL. You should attach and start transactions explicitly.

For more information about the Encina calls, see the Encina documentation.

Using SQL Context Structures

You use a **context structure**, a host language structure, to pass information about the distributed transaction. The *Oracle Rdb7 Guide to Distributed Transactions* describes the **context structure** in more detail.

If you are using an XA transaction manager, you declare the context structure as shown in the following example:

```
struct {
        int version;
        int type;
        int length;
        int value;
        int end;
        } xacontext = {1,2,4,1,0};
```

You must associate the context structure with most *executable* SQL statements. This is true whether you use the SQL module language or precompiled SQL, although the method you use to associate the context structure with SQL statements differs depending on the SQL interface you choose.

To use context structures with SQL module language, take the following actions:

- Declare a context structure in the host language program.
- Pass the context structure to *most* SQL module procedures.

  You pass the context structure to those procedures that contain executable SQL statements (except for those listed in the *Oracle Rdb7 Guide to Distributed Transactions*). For example, to pass the context structure from a C language program to an SQL module procedure called update_pers, use the following code:

  ```
  update_pers( &sqlcode, &context_struc)
  ```

- Process the SQL source files using the `-ctx` qualifier on the SQL module processor command line.

  You can use the following arguments to the `-ctx` qualifier:

  – `all`
  – `none`
  – `"(proc_name,...)"`

To use context structures with precompiled SQL, you must take the following actions:

- Declare a context structure in the program.
- Add the USING CONTEXT clause to *most* executable SQL statements that are involved in the distributed transaction.

  For example, to pass the context structure to a statement that opens a cursor, use the following code:

  ```
  EXEC SQL USING CONTEXT :context_struc OPEN CURSOR1
  ```

See the *Oracle Rdb7 Guide to Distributed Transactions* for more information about context structures, including which statements need the context structure.

The Sample Program

A sample program demonstrating how to use Encina with Oracle Rdb is supplied with the kit in the directory /usr/lib/dbs/sql/v70/examples. The sample program is in two files, encinasample.sc and encinainit.c. The following excerpt, using the Encina toolkit, shows the order of initialization:

```
/*
** Register the databases with the transaction manager.
*/
    MakeInfo( "personnel", persopeninfo );           1
    tstat = tmxa_RegisterRMI(
                persopeninfo, "",
                &xaordbsw,
                TMXA_SERIALIZE_ALL_XA_OPERATIONS,
                &persID );

    CHK_STAT("tmxa_RegisterRMI for PERSONNEL", tstat);

    MakeInfo( "mf_personnel", mfopeninfo );          2
    tstat = tmxa_RegisterRMI(
                mfopeninfo, "",
                &xaordbsw,
                TMXA_SERIALIZE_ALL_XA_OPERATIONS,
                &mfID );

    CHK_STAT("tmxa_RegisterRMI for MF_PERSONNEL", tstat);
```

```
/*
** Initialize the transaction manager.
*/
    if (!Init_TM()) exit(1);
/*
** Force the attaches to happen now.  The attaches must happen before
** the XA transaction starts.
*/
exec sql connect 'alias PERS, alias MFPERS';         3
```

The following points explain the callouts in the preceding example:

1 Register the personnel database with the transaction manager.

2 Register the mf_personnel database with the transaction manager.

3 Attach to the databases using the SQL CONNECT statement. The DECLARE ALIAS statements (not shown here) map the aliases to the file names.

The following excerpt starts a transaction and performs an update operation:

```
/*
** Start the XA transaction.  This must happen before the SQL transaction
** starts.
*/
    status = tx_begin();                             1

    if ( status != TX_OK)
        {
        printf ("tx_begin failed, status %d\n", status );
        exit(1);
        }

/*
** Start the SQL transaction.                        2
*/
exec sql using context :xacontext set transaction
     on PERS   using (READ WRITE RESERVING PERS.EMPLOYEES FOR SHARED WRITE)
 and on MFPERS using (READ WRITE RESERVING MFPERS.EMPLOYEES FOR SHARED WRITE);

exec sql using context :xacontext                    3
         update PERS.EMPLOYEES
         set    ADDRESS_DATA_1       = :street,
                ADDRESS_DATA_2       = :address_data,
                CITY                 = :town,
                STATE                = :state,
                POSTAL_CODE          = :postal_code
        where EMPLOYEE_ID = :employee_id ;
    if (SQLCA.SQLCODE != 0)
      {  succeed = FALSE;
         handle_error("updating PERS.EMPLOYEES");
      } ;
```

```
/*
** Announce the success or failure of the modify operation, and
** commit or roll back the transaction accordingly.
*/
    if (succeed == TRUE)                                  4
        {
        printf ("Update operation succeeded\n");
        status = tx_commit();
        if (status != TX_OK)
            {
            printf ("tx_commit failed, status = %d\n", status );
            }
        }
    else
        {
        printf ("Update operation failed\n");
        status = tx_rollback();

        if (status != TX_OK)
            {
            printf ("tx_rollback failed, status = %d\n", status );
            }
        }
```

The following points explain the callouts in the preceding example:

1 Start the XA transaction, using the Encina tx_begin call.

2 Start the SQL transaction, using the SET TRANSACTION statement. Note that the statement contains the USING CONTEXT clause.

3 Perform operations in the distributed transaction. Note that the statement contains the USING CONTEXT clause.

4 Roll back or commit the transaction using the Encina tx_rollback or tx_ commit calls.

Building and Running the Sample Program

Use the following commands to compile and link the sample program for Encina:

```
setenv sql_sample /usr/lib/dbs/sql/v70/examples
#
sqlpre -l cc="-O0 -g -I/opt/encina/include" -s '-msgvec -conn' \
-o encinasample.o $sql_sample/encinasample.sc
#
cc -O0 -g -I/opt/encina/include \
-o encinainit.o -c $sql_sample/encinainit.c
#
cc -g -o encinasample encinasample.o encinainit.o \
-lEncina -lEncServer -ldce \
-L/usr/lib/dbs/shlib -lsql -lrdbshr -lcosi -lots
```

Before executing the program, you must create a special log file for Encina. Issue the following commands from a *root* account:

```
echo x | dd of=enclogvol seek=8k
chown {yourname} enclogvol
chmod 700 enclogvol
```

Then, define the following three environment variables to specify the location of program's log files:

```
setenv RDBENCINA_LOG_VOL {path to the file created } /enclogvol
setenv RDBENCINA_LOG_DIR {path to a directory to get log file}
setenv RDBENCINA_RESTART {log directory}/r1:{log directory}/r2
```

Debugging XA Applications

To help debug and trace XA transactions, you can use the configuration parameter SQL_XA_TRACE in your .dbsrc configuration file. When you set this parameter equivalent to "TRUE", Oracle Rdb supplies trace information that makes it easier to debug your applications. (For information about the .dbsrc configuration file, see the *Migrating Oracle Rdb7 Databases and Applications to Digital UNIX*.)

1.8.5.5 Recovering from Unresolved Transactions

To recover from unresolved distributed transactions, use the following Oracle RMU commands:

- RMU Dump Users command with the State=Blocked qualifier
- RMU Resolve command
- RMU Recover Resolve command
- RMU Dump After_Journal command with the State=Prepared qualifier

For more information about these commands, see the *Oracle RMU Reference Manual*. For more inforamtion about distributed transactions, see the *Oracle Rdb7 Guide to Distributed Transactions*, which describes the concepts of distributed transactions and explains how to use distributed transactions on Digital UNIX.

1.8.5.6 Compliance Information

Oracle Rdb V7.0 supports only the Encina transaction manager from Transarc. Other transaction managers, such as TUXEDO from Novell, use some parts of the XA specification that Oracle Rdb does not support yet. Therefore, Oracle Rdb can only claim *partial* compliance with the XA specification for V7.0.

The XA specification requires that resource managers publish the following information:

- **xa_switch_t structure name**: `xaordbsw`

  This structure contains entry points and other information about the resource manager.

- **resource manager name**: `Oracle_Rdb_XA`

  This is the Oracle Rdb resource manager name within the xa_switch_t structure.

- **close info string**

  The close information string used by xa_close is ignored and it may be null.

- **open info string**

  The open information string is required by xa_open. The open string has a maximum size of 256 characters and it must be in the format described in Database References in Section 1.8.5.4.

- **linking**

  For an application to use XA transactions, the application must be linked with the standard shareable library /usr/shlib/librdbshr.so.

- **SQL semantics**

Using the SET TRANSACTION or DECLARE TRANSACTION statement with the BATCH UPDATE clause returns an error at run time. Because batch-update transactions do not write to recovery-unit journal files, batch-update transactions cannot be rolled back. XA requires that a transaction can be rolled back.

1.8.5.7 Optional Features

XA resource managers also have the option of implementing various XA features. The following describes which optional features are supported and not supported by Oracle Rdb:

- **Protocol optimizations**

  Oracle Rdb supports the read-only optimization, which allows Oracle Rdb to not participate in the two-phase commit protocol when a database is being accessed as read-only.

- **Association migration**

  Oracle Rdb does not support association migration. Association migration is a means by which a transaction manager may resume a suspended branch association in another branch.

- **Dynamic registration**

  Oracle Rdb does not support dynamic registration. Dynamic registration allows a resource manager to determine when it will participate in a two-phase commit transaction.

- **Asynchrony**

  Oracle Rdb does not support asynchronous XA calls.

- **Heuristics**

  Oracle Rdb does not support heuristic decision-making. This optional feature allows a resource manager that has prepared to commit a transaction branch the ability to commit or roll back its work independently of the transaction manager.

♦

1.9 New Features in SQL

This section summarizes new and changed features for the SQL interface to Oracle Rdb. These features include:

- Vertical partitioning

  You can now partition a table vertically as well as horizontally. When you partition a table horizontally, you divide the rows of the table among storage areas according to data values in one or more columns. Then, a given storage area contains only those rows whose column values fall within the range that you specify. When you partition a table vertically, you divide the columns of the table among storage areas. Then, a given storage area contains only some of the columns of a table. Consider using vertical partitioning when you know that access to some of the columns in a table is frequent, but that the access to other columns is occasional.

  For more information, see the *Oracle Rdb7 Guide to Database Design and Definition* and the *Oracle Rdb7 SQL Reference Manual*.

- Strict partitioning

You can now specify whether a partitioning key for a storage map is updatable or not updatable. If you specify that the key is not updatable, Oracle Rdb retrieval performance improves because Oracle Rdb can use the partitioning criteria when optimizing the query.

For more information, see the *Oracle Rdb7 Guide to Database Design and Definition* and the *Oracle Rdb7 SQL Reference Manual*.

- Creating a default storage area

  You can separate user data from the system data, such as the system tables, by using the DEFAULT STORAGE AREA clause of the CREATE DATABASE or IMPORT statements. This clause specifies that all user data and indexes that are not mapped explicitly to a storage area be stored in the default storage area. For more information, see the *Oracle Rdb7 Guide to Database Design and Definition* and the *Oracle Rdb7 SQL Reference Manual*.

- Dropping a storage area with a cascading delete

  You can specify that Oracle Rdb drop a storage area with a cascading delete. When you do, Oracle Rdb drops database objects referring to the storage area. For more information, see the *Oracle Rdb7 Guide to Database Design and Definition* and the *Oracle Rdb7 SQL Reference Manual*.

- Altering the RDB$SYSTEM storage area

  You can now alter the RDB$SYSTEM storage area using the ALTER STORAGE AREA clause of the ALTER DATABASE statement. You can use the ALTER STORAGE AREA clause to alter the following characteristics:

  ALLOCATION IS number-pages PAGES
  extent-params
  CACHE USING cache-name
  NO ROW CACHE
  SNAPSHOT ALLOCATION IS snp-pages PAGES
  SHAPSHOT EXTENT
  CHECKSUM CALCULATION
  SNAPSHOT CHECKSUM CALCULATION

  For more information, see the *Oracle Rdb7 Guide to Database Design and Definition* and the *Oracle Rdb7 SQL Reference Manual*.

- Importing and exporting a view based on a system table

  You can now import and export databases that contain views based on system tables. In previous versions, attempting to export a database with views based on system tables resulted in the error `SQL-F-FLDNOTBCK`.

- Moving optional system tables

  You can move optional system tables from the RDB$SYSTEM storage area or the default storage area to other storage areas by using the CREATE STORAGE MAP statement and then the ALTER STORAGE MAP statement.

  You can move the following system tables:

  RDB$CATALOG_SCHEMA (Optional table for multischema databases)
  RDB$CHANGES (Optional table for the Replication Option for Rdb)
  RDB$CHANGES_MAX_TSER (Optional table for the Replication Option for Rdb)
  RDB$SYNONYMS (Optional table for multischema databases)
  RDB$TRANSFERS (Optional table for the Replication Option for Rdb)

RDB$TRANSFER_RELATIONS (Optional table for the Replication Option for Rdb)
RDB$WORKLOAD (Optional table for workload collection)

See the *Oracle Rdb7 Guide to Database Design and Definition* for a description of how to move these areas and the restrictions involved.

- Creating storage maps for tables with data

  In previous versions, you could not create a storage map for tables that contained data. Now, if the table is located in either the RDB$SYSTEM storage area or the default storage area and has no storage map, you can create a storage map for the table. For more information, see Section 3.2.25 and the *Oracle Rdb7 Guide to Database Design and Definition*.

- Creating outlines for stored functions

  The ON FUNCTION clause of the CREATE OUTLINE statement lets you create an outline directly on a stored function. In addition, you can specify the USING clause with the ON PROCEDURE and ON FUNCTION clauses. For more information, see the *Oracle Rdb7 SQL Reference Manual*.

- New FROM clause for CREATE OUTLINE

  The process for creating outlines has been simplified with new syntax. You can now specify the SELECT statement within the FROM clause of the CREATE OUTLINE statement for which you need an outline. For more information, see the *Oracle Rdb7 SQL Reference Manual*.

- Freezing data definitions

  You can ensure that the data definitions of your database do not change by using the METADATA CHANGES ARE DISABLED clause of the ALTER DATABASE, CREATE DATABASE, or IMPORT statements. For more information, see the *Oracle Rdb7 Guide to Database Design and Definition* and the *Oracle Rdb7 SQL Reference Manual*.

- Modifying the database buffer size

  You can now modify the database buffer size by using the BUFFER SIZE clause in the ALTER DATABASE statement. In previous versions, you could specify the clause only in the CREATE DATABASE statement. For more information, see the *Oracle Rdb7 Guide to Database Design and Definition* and the *Oracle Rdb7 SQL Reference Manual*.

- Specifying how a database opens when you create the database

  You can specify whether a database opens automatically or manually when you create the database. In previous versions, you could specify the OPEN IS clause only in the ALTER DATABASE statement. For more information, see the *Oracle Rdb7 Guide to Database Design and Definition* and the *Oracle Rdb7 SQL Reference Manual*.

- Specifying how long to wait before closing a database

  You can specify how long Oracle Rdb waits before closing the database, by using the WAIT n MINUTES FOR CLOSE clause. For more information, see the *Oracle Rdb7 SQL Reference Manual*.

- Extending the allocation of storage areas

You can now manually force a storage area to extend by using the ALLOCATION IS clause of the ALTER STORAGE AREA clause. For more information, see the *Oracle Rdb7 Guide to Database Design and Definition* and the *Oracle Rdb7 SQL Reference Manual*.

- Quickly deleting data in tables

  If you want to quickly delete the data in a table, but you want to maintain the metadata definition of the table (perhaps to reload the data into a new partitioning scheme), you can use the TRUNCATE TABLE statement. For more information, see the *Oracle Rdb7 Guide to Database Design and Definition* and the *Oracle Rdb7 SQL Reference Manual*.

- Creating temporary tables

  You can create temporary tables to store temporary results only for a short duration, perhaps to temporarily store the results of a query so that your application can act on the results of that query. The data in a temporary table is deleted at the end of an SQL session. For more information, see the *Oracle Rdb7 Guide to Database Design and Definition* and the *Oracle Rdb7 SQL Reference Manual*.

- New logical name and configuration parameter for use with temporary tables

  The logical name RDMS$TTB_HASH_SIZE or configuration parameter RDB_TTB_HASH_SIZE sets the size of the hash table used for temporary tables. If the logical name or configuration parameter is not defined, Oracle Rdb uses a default value of 1249.

  If you expect that that temporary tables will be large (that is, 10K or more rows), use this logical name or configuration parameter to adjust the hash table size to avoid long hash chains. Set the value to approximately 1/4 of the expected maximum number of rows for each temporary table. For example, if a temporary table will be populated with 100,000 rows, define this logical name or configuration parameter to be 25000. If there are memory constraints on your system, you should define the logical name or configuration parameter to be no higher than this value (1/4 of the expected maximum number of rows).

- Removing the links to the repository

  You can remove the link between the repository and database but still maintain the data definitions in both places, using the DICTIONARY IS NOT USED clause of the ALTER DATABASE statement.

  For more information, see the *Oracle Rdb7 Guide to Database Design and Definition* and the *Oracle Rdb7 SQL Reference Manual*.

- Specifying the location of the recovery journal

  You can specify the location of the recovery journal using the RECOVERY JOURNAL (LOCATION IS 'directory-spec') clause with the ALTER DATABASE, CREATE DATABASE, and IMPORT statements.

  For more information, see the *Oracle Rdb7 SQL Reference Manual*.

- Specifying an edit string in an .aij backup file name

  You can specify whether the backup file name includes an edit string with the EDIT STRING clause of the ALTER DATABASE statement.

  For more information, see the *Oracle Rdb7 SQL Reference Manual*.

- Increasing the fanout factor for adjustable lock granularity

Adjustable lock granularity for previous versions of Oracle Rdb defaulted to a count of 3. This means that the lock fanout factor was (10, 100, 1000). As databases grow larger, it is necessary to allow these fanout factors to grow to reduce lock requirements for long queries. You can now change the fanout factor by specifying the COUNT IS clause with the ADJUSTABLE LOCK GRANULARITY IS ENABLED clause.

For more information, see the *Oracle Rdb7 Guide to Database Design and Definition* and the *Oracle Rdb7 SQL Reference Manual*.

- Specifying detected asynchronous prefetch with a threshold value

  Detected asynchronous prefetch can significantly improve performance by using heuristics to determine if an I/O pattern is sequential even if the operation is not actually performing sequential I/O. For example, when fetching a LIST OF BYTE VARYING column, the heuristics detect that the pages being fetched are sequential and fetch ahead asynchronously to avoid wait times when the page is really needed.

  For more information, see the *Oracle Rdb7 SQL Reference Manual*.

- Setting debug flags using SQL

  SQL supports a new SET FLAGS statement in interactive and dynamic SQL and a SHOW FLAGS statement in interactive SQL. The SET FLAGS statement lets you enable and disable the database systems debug flags during execution. For more information, see the *Oracle Rdb7 SQL Reference Manual* and the *Oracle Rdb7 Guide to SQL Programming*.

- Holding cursors open across transactions

  SQL cursors can now remain open across transaction boundaries. The WITH HOLD clause of the DECLARE CURSOR statement indicates that the cursor remains open after the transaction ends. A holdable cursor that has been held open retains its position when a new SQL transaction is begun.

  You can also specify the attributes of the holdable cursor as a database default using the SET HOLD CURSORS statement.

  For more information, see the *Oracle Rdb7 SQL Reference Manual* and the *Oracle Rdb7 Guide to SQL Programming*.

- External routine enhancements

  SQL now provides external procedures and allows external routines to contain SQL statements to bind to new schema instances and perform database operations. **External routines** are external functions or external procedures that are written in a 3GL language such as C or Fortran, linked into a shareable image, and registered in a database schema.

  External routine activation, execution, and exception handling is controlled by a new executor manager process. External routines are available on all platforms.

  For more information, see the *Oracle Rdb7 Guide to SQL Programming* and the *Oracle Rdb7 SQL Reference Manual*.

- Creating stored functions

  In addition to defining stored procedures, you can now define stored functions using the CREATE MODULE statement. A **stored function** is a set of operations performed on an Oracle Rdb database by one or more SQL statements. It accepts a set of input parameters and returns a single result. You invoke a stored function by using the function name in a value expression.

See the *Oracle Rdb7 SQL Reference Manual* and the *Oracle Rdb7 Guide to SQL Programming* for more information regarding stored functions.

- Returning the value of a stored function

  SQL provides the RETURN statement, which returns the result of a stored function. See the *Oracle Rdb7 SQL Reference Manual* and the *Oracle Rdb7 Guide to SQL Programming* for more information.

- Cascading delete for modules

  The DROP MODULE CASCADE statement lets you drop a module and invalidates any objects that refer to it. See the *Oracle Rdb7 SQL Reference Manual* and the *Oracle Rdb7 Guide to SQL Programming* for more information.

- Dropping functions and procedures

  You can now drop external procedures and stored functions and procedures. See the *Oracle Rdb7 SQL Reference Manual* and the *Oracle Rdb7 Guide to SQL Programming* for more information.

- Using the CALL statement in a compound statement

  You can now use the CALL statement within a compound statement, thus in a stored procedure or function, to call another stored procedure. You can also use the CALL statement to invoke external procedures.

  For more information, see the *Oracle Rdb7 Guide to SQL Programming* and the *Oracle Rdb7 SQL Reference Manual*.

- New SIGNAL statement

  SQL now provides a new SIGNAL statement for use within a compound statement. The SIGNAL statement accepts a single character value expression, which is used as the SQLSTATE. When Oracle Rdb encounters a SIGNAL statement, the current routine and all calling routines are terminated and the signaled SQLSTATE is returned to the application.

  For more information, see the *Oracle Rdb7 SQL Reference Manual* and the *Oracle Rdb7 Guide to SQL Programming*.

- Specifying the DEFAULT clause, CONSTANT clause, and UPDATABLE clause when declaring variables within compound statements

  You can use the DEFAULT clause to specify the default value of a variable to be any value expression including subqueries, conditional, character, date /time, and numeric expressions. Additionally, the variable can now inherit the default from the named domain.

  The CONSTANT clause changes the variable into a declared constant, which cannot be updated. The UPDATABLE clause allows a variable to be updated.

  For more information, see the *Oracle Rdb7 SQL Reference Manual* and the *Oracle Rdb7 Guide to SQL Programming*.

- Obtaining the connection name and calling routine name using the GET DIAGNOSTICS statement

  You can now obtain the current connection name in a variable or parameter from within a compound statement using the GET DIAGNOSTICS statement.

  You can also obtain the calling routine name using the GET DIAGNOSTICS statement.

For more information, see the *Oracle Rdb7 SQL Reference Manual* and the *Oracle Rdb7 Guide to SQL Programming*.

- EXTERNAL keyword for INCLUDE SQLCA statement

  You can now declare an external reference to the SQLCA structure when you use the SQL precompiler with the C language by using the optional EXTERNAL keyword in the INCLUDE SQLCA statement.

  For more information, see the *Oracle Rdb7 Guide to SQL Programming* and the *Oracle Rdb7 SQL Reference Manual*.

- Two new basic predicates for inequality comparisons

  The inequality predicates are ^= and !=.

  The != predicate is available only if you set your dialect to ORACLE LEVEL1. See the *Oracle Rdb7 SQL Reference Manual* for more information on basic predicates.

- Changes to CURRENT_USER function

  In V7.0, before any invoker's rights routines are called, the CURRENT_USER function is established as identical to the SESSION_USER function. As each routine is called, it either inherits the value of the authorization from the caller or, in the case of a definer's rights routine, derives it from the module AUTHORIZATION clause. Therefore, the CURRENT_USER function returns the authorization of the last definer's rights routine in the call chain.

  For more information on CURRENT_USER, see the *Oracle Rdb7 SQL Reference Manual*.

- Specifying the new dialect ORACLE LEVEL1

  You can now specify the ORACLE LEVEL1 dialect for the interactive SQL and dynamic SQL environments. This dialect is similar to the SQL92 dialect. For more information, see the *Oracle Rdb7 SQL Reference Manual*.

OpenVMS VAX OpenVMS Alpha

- Specifying C_PROTOTYPES=file-name qualifier for SQL module language

  The SQL module language C_PROTOTYPES qualifier now accepts a file name. See the *Oracle Rdb7 SQL Reference Manual* for more information. ♦

Digital UNIX

- Editing in interactive SQL

  On Digital UNIX, you can use the EDIT statement within interactive SQL. It works similarly to the SQL EDIT statement on OpenVMS. For more information, see the *Migrating Oracle Rdb7 Databases and Applications to Digital UNIX* and the *Oracle Rdb7 SQL Reference Manual*. ♦

Digital UNIX

- Support for Pascal and FORTRAN on Oracle Rdb for Digital UNIX

  Oracle Rdb for Digital UNIX now supports the DEC FORTRAN and DEC Pascal languages for the SQL precompiler and the SQL module processor. ♦

Digital UNIX

- New command line qualifier for precompiled SQL

  The –extend_source command line qualifier allows the SQL precompiler to view 132 columns of FORTRAN source rather than the default of 72 columns.

  See the *Oracle Rdb7 SQL Reference Manual* and the *Oracle Rdb7 Guide to Database Design and Definition* for more information. ♦

- New Pascal data types

  SQL now supports the following built-in Pascal data types:

  – INTEGER8

  - INTEGER16
  - INTEGER32
  - INTEGER64

- The Shift_JIS character set

  Oracle Rdb includes support for the Shift_JIS character set, a mixed multi-octet character set.

  For more information, see the *Oracle Rdb7 SQL Reference Manual*.

- Setting transaction modes

  You can enable or disable transaction modes on the CREATE DATABASE or ALTER DATABASE statements, giving you more control over the database environment.

  For more information, see the *Oracle Rdb7 SQL Reference Manual*.

- Enhancements for the SQL SHOW statement

  The SQL SHOW statement displays the new features affecting data definition, stored routines, and external routines.

  For more information, see the *Oracle Rdb7 SQL Reference Manual*.

- The keyword ROWID

  You can use the keyword ROWID as a synonym for the keyword DBKEY. For more information, see the *Oracle Rdb7 SQL Reference Manual*.

- COUNT function enhancements

  You can now specify:

  - COUNT (*)
  - COUNT (value-expr)
  - COUNT (DISTINCT value-expr)

  For more information, see the *Oracle Rdb7 SQL Reference Manual*.

- Oracle7 SQL functions

  New SQL functions are added to the Oracle Rdb SQL interface for compatibility with Oracle7 SQL. Complete descriptions of these functions can be found in the *Oracle7 Server SQL Language Reference Manual*.

  The following new functions are built in to the SQL interface of Oracle Rdb V7.0:

  - CONCAT—Concatenates two strings. CONCAT is functionally equivalent to the concatenation operator (| |).
  - CONVERT—Converts a character string to the specified character set.
  - DECODE—Compares an expression to a search value until a match is found.
  - SYSDATE—Returns the current date and time. SYSDATE is a synonym for CURRENT_TIMESTAMP.

  In addition to the listed built-in functions, you can install many other functions that emulate Oracle7 functions by using a script in interactive SQL. For more information on these optional functions and on the listed built-in functions, see the *Oracle Rdb7 SQL Reference Manual*.

- SQL header file for use with C programs

  You can now use the sql_sqlda.h header file in C language programs to obtain definitions of the SQLDA and SQLDA2 structures. Previously, you could only obtain definitions of these structures by using the SQL precompiler statement INCLUDE SQLDA or INCLUDE SQLDA2. You can now use the sql_sqlda.h header file in an include directive in a C language source module.

  For more information, see the *Oracle Rdb7 Guide to SQL Programming*.

- SQL header file to eliminate DEC C informational messages

  SQL provides a header file, sql_rdb_headers.h to eliminate informational messages by providing prototypes for explicitly called SQL routines. For more information, see the *Oracle Rdb7 Guide to SQL Programming*.

1.10 New Features in Oracle RMU

This section summarizes new and changed features for the Oracle RMU interface to Oracle Rdb. These features include:

- Hot Standby database option

  For information, see Section 1.6.

- Temporary work files for AIJ rollforward operations can be directed to a new location

  You can redirect the temporary work files for AIJ rollforward operations and the database recovery (DBR) redo operations to a different disk and directory location than the default. Do this by assigning a different directory to the new RDM$BIND_AIJ_WORK_FILE and RDM$BIND_DBR_WORK_FILE logical names on OpenVMS systems or the RDM_BIND_AIJ_WORK_FILE and RDM_BIND_DBR_WORK_FILE configuration parameters on Digital UNIX systems. This can be helpful in alleviating I/O bottlenecks in the default location.

  See the sections on the RMU Recover and RMU Optimize After_Journal commands in the *Oracle RMU Reference Manual* for details.

- Tape loader or stacker alert

  By default, if a tape device has a loader or stacker, Oracle RMU should recognize this fact. However, occasionally Oracle RMU does not recognize that a tape device has a loader or stacker. Therefore, when the first tape fills or has been read (depending on the operation), Oracle RMU issues a request to the operator for the next tape, instead of requesting the next tape from the loader or stacker. Similarly, sometimes Oracle RMU behaves as though a tape device has a loader or stacker when actually it does not.

  The new Media_Loader and Nomedia_Loader qualifiers allow you to alert Oracle RMU to the presence or absence of a tape loader or stacker. These qualifiers are valid with the RMU Backup, RMU Backup After_Journal, RMU Dump After_Journal, RMU Dump Backup, RMU Optimizer After_Journal, RMU Recover, RMU Recover Resolve, RMU Restore, and RMU Restore Only_Root commands.

  See the *Oracle RMU Reference Manual* for details.

- RdbAlter Area . . . Page command enhancement

The RdbAlter Area . . . Page command now allows you to specify a snapshot area as the area-name parameter. This, in turn, allows you to use the RdbAlter Display command and RdbAlter Deposit command with snapshot files.

See the *Oracle RMU Reference Manual* for details.

- RMU Analyze Cardinality command replacement

  The RMU Analyze Cardinality command has been replaced with the new RMU Collect Optimizer_Statistics command. The new command offers improved functionality over the RMU Analyze Cardinality Command. The RMU Analyze Cardinality command is now deprecated.

  See the description of the new command within this list and in the *Oracle RMU Reference Manual* for details.

- RMU Analyze Index and RMU Analyze Placement command enhancements

  The following enhancements have been made to the RMU Analyze Index and Analyze Placement commands:

  - These commands display statistics on the new sorted ranked indexes created with SQL.
  - A new qualifier, Transaction_Type, is now supported by these commands. This new qualifier allows you to specify the type of transaction mode Oracle RMU should use when performing the analyze operation, or allows you to specify that Oracle RMU should determine the transaction type based on whether or not snapshot areas have been disabled.

  See the *Oracle RMU Reference Manual* for details.

- RMU Backup command enhancements

  OpenVMS VAX OpenVMS Alpha

  - Support for multiprocess backup operations to tape

    You can now specify a multiprocess RMU Backup command, referred to as a **parallel backup** for backup operations to tape. This new feature uses multiple, multithreaded processes to perform a database backup. A parallel backup operation significantly improves the performance of large database backup operations on SMP systems and VMSclusters. The new Oracle RMU qualifiers to support parallel backup are the Execute and Parallel qualifiers.

  - Support for generating a parallel backup plan file

    Oracle RMU can generate a plan file for your parallel backup operation that contains information about the parallel backup operation and work instructions for the multiple processes. When you specify the plan file as a parameter to the new RMU Backup Plan command, Oracle RMU executes the backup operation per the specifications included in the plan file. ♦

    The new RMU Backup qualifier to support backup plan generation is the List_Plan qualifier.

  - Support for generating an options file that can be used with the RMU Restore command

    A new qualifier, Restore_Options, has been added to the RMU Backup command. This qualifier directs Oracle RMU to generate an options file that you can specify as the parameter to the Options qualifier of the RMU Restore command.

- Support for accepting tape labels

  The RMU Backup command now allows you to specify a qualifier that directs Oracle RMU to keep the label it finds on a tape during a backup operation even if that label does not match the default label or that specified with the Label qualifier. The new qualifier is the Accept_Label qualifier.

See the *Oracle Rdb7 Guide to Database Maintenance* and the *Oracle RMU Reference Manual* for details.

• RMU Backup After_Journal command enhancements

The following new features have been added to the RMU Backup After_ Journal command:

- The RMU Backup After_Journal command now allows you to specify the maximum time the .aij backup file operation waits for the quiet-point lock during online backup operations. The new qualifier to support this feature is the Lock_Timeout qualifier.
- A Sequence option to Edit_Filename qualifier has been added. This option is synonymous to the Vno option. Either option can be used and has the same effect. The synonym has been added to more accurately reflect the semantics of the option. Both the Vno and Sequence options specify that the .aij sequence number is to be used in the .aij backup file edit string.
- OpenVMS VAX OpenVMS Alpha Two new global process symbols, RDM$AIJ_ENDOFFILE and RDM$AIJ_ FULLNESS, are set when the RMU Backup After_Journal command completes. RDM$AIJ_ENDOFFILE contains the end of file block number for the current .aij file. RDM$AIJ_FULLNESS contains the percent fullness of the current .aij file. ♦
- The RMU Backup After_Journal command now allows you to specify a qualifier that directs Oracle RMU to keep the label it finds on a tape during an .aij backup operation even if that label does not match the default label or that specified with the Label qualifier. The new qualifier is the Accept_Label qualifier.

See the *Oracle Rdb7 Guide to Database Maintenance* and the *Oracle RMU Reference Manual* for details.

OpenVMS VAX OpenVMS Alpha

• New RMU Backup Plan command

This command allows you to execute a plan file generated with the new List_Plan qualifier of the RMU Backup command.

See the *Oracle Rdb7 Guide to Database Maintenance* and the *Oracle RMU Reference Manual* for details. ♦

• RMU Checkpoint command enhancement

The RMU Checkpoint command now supports the Wait and Nowait qualifiers. These qualifiers allow you to specify whether or not the system prompt is to be returned before the checkpoint operation completes.

See the *Oracle RMU Reference Manual* for details.

• New RMU Collect Optimizer_Statistics, Delete Optimizer_Statistics, Insert Optimizer_Statistics, and Show Optimizer_Statistics commands

These new commands allow you to maintain and manage three types of statistics for the optimizer. The three types of statistics are cardinality, workload, and storage. The cardinality statistics include table cardinality, index cardinality, and index prefix cardinality. The workload statistics are duplicity factor and null factor, which are based on interesting column groups derived from a query workload. The storage statistics are row clustering factor, index depth (sorted index only), key clustering factor, and data clustering factor.

The optimizer uses these statistics to improve the estimated cost and cardinality of various solutions tried during query optimization. This increases the probability that the optimizer will generate an optimal query solution.

The workload statistics can be collected only after workload information has been gathered and stored into a system table by the optimizer. You enable the collection of workload information by the optimizer using the WORKLOAD COLLECTION IS ENABLED clause of the SQL CREATE DATABASE or SQL ALTER DATABASE statement.

The RMU Collect Optimizer_Statistics command allows you to specify whether statistics are collected for all or specified tables and indexes. The RMU Show Optimizer_Statistics command displays the existing statistics for specified tables or indexes. The RMU Delete Optimizer_Statistics command removes entries from statistics collection. The RMU Insert Optimizer_Statistics command replaces workload statistics that you deleted previously.

See the *Oracle RMU Reference Manual*, the *Oracle Rdb7 SQL Reference Manual*, and the *Oracle Rdb7 Guide to Database Performance and Tuning* for details.

- RMU Convert command enhancements

  The following changes have been made to the RMU Convert command:

  - The RMU Convert command no longer supports conversion of pre-V5.1 databases. If you have a database whose structure is pre-V5.1, first convert it to V5.1, V6.0, or V6.1, and then convert the database to V7.0. See the *Oracle Rdb7 Installation and Configuration Guide* for details.
  - When a database from an earlier version is converted to V7.0 or higher, the RMU Convert command collects and stores the index prefix cardinalities for each multisegmented sorted index in the database.

    Because the collection of index prefix cardinality information upon conversion of a large database can take a considerable amount of time, the RMU Convert command issues a new prompt that gives you the option of either computing or estimating the index prefix cardinalities. For more information, see Section 1.10.1.

  See the *Oracle RMU Reference Manual* for details.

- RMU Copy command enhancement

  A new qualifier, Transaction_Mode, sets the allowable transaction modes for the database root file created by this command. For more information, see Section 4.4.1.

- RMU Dump command enhancements

  The following enhancements have been made to the RMU Dump command:

  - Restore_Options qualifier

    A new qualifier, Restore_Options, directs Oracle RMU to generate an options file that can be specified as the parameter to the Options qualifier of the RMU Restore command.

  - Header qualifier

    The Header qualifier has been enhanced such that you can now provide an options list. The options list allows you to limit the output from the Dump command to specific areas of interest.

  See the *Oracle RMU Reference Manual* for details.

- RMU Dump After_Journal command enhancement

  A new qualifier, Option=[No]Statistics, allows you to specify that Oracle RMU display statistics on how frequently database pages are referenced by the data records in the .aij file. In addition, if the database root file is available, Oracle RMU displays a recommended value for the RMU Recover command's Aij_Buffers=n qualifier.

  See the *Oracle RMU Reference Manual* for details.

- RMU Dump Backup command enhancements

  - A new qualifier, Restore_Options, directs Oracle RMU to generate an options file that can be specified as the parameter to the Options qualifier of the RMU Restore command.
  - Now when you specify the Root, Full, or Debug option, Oracle RMU includes database backup header information in the dump. This header information provides the backup file name and the backup file database version. The backup file database version is the version of Oracle Rdb that was executing at the time the backup file was created.

  See the *Oracle RMU Reference Manual* for details.

- RMU Extract command enhancements

  The following enhancements have been made to the RMU Extract command:

  - New Items=Procedures option

    A new option, Procedures, is supported for use with the Items qualifier. This new option allows you to direct Oracle RMU to extract external procedures.

  - New Items=Import option

    A new parameter, Import, is supported for use with the Items qualifier. This new parameter allows you to direct Oracle RMU to generate an SQL or RDO IMPORT script.

  - New default behavior for the Options=Dictionary_References qualifier.

    If the Options=Dictionary_References qualifier or the Options=Nodictionary_References qualifier is not specified, Oracle RMU examines the RDB$RELATIONS and RDB$FIELDS system tables to determine whether or not any domains or tables refer to the repository (data dictionary). If references are made to the repository, the Options=Dictionary_References qualifier is the default. Otherwise, Oracle RMU assumes that the repository is not used, and the default is the Options=Nodictionary_References qualifier.

  See the *Oracle RMU Reference Manual* for details.

- RMU Load command enhancements

The RMU Load command now supports the following:

– Parallel load operations

  A new qualifier, Parallel, allows you to specify a multiprocess RMU Load command (referred to as a parallel load). A parallel load operation can be used to increase the speed of a large load operation.

– Setting when and if constraints are evaluated

  Three new qualifiers, Constraints, Constraints=Deferred, and Noconstraints, allow you to determine when or if constraints are evaluated during a load operation.

– Support for deferring index updates

  A new qualifier, Defer_Index_Updates, allows you to specify that non-unique indexes other than those that define the placement information for data in a storage area will not be rebuilt until commit time.

– Generating a load plan file

  A new qualifier, List_Plan, allows you to generate a file containing all the information needed by Oracle RMU to execute a load procedure. This file is called a load plan file. You can determine whether the load plan file is executed by issuing the new Execute or Noexecute qualifier. If you specify the Noexecute qualifier on the RMU Load command, you can process the plan file later by issuing the new RMU Load Plan command.

– Specifying the number of rows sent between processes in a single I/O request

  A new qualifier, Row_Count, allows you to specify the number of rows that are sent between processes in a single I/O request during a load operation. This qualifier is designed primarily for use with Oracle Rdb for Digital UNIX databases.

OpenVMS VAX OpenVMS Alpha

– Loading security audit records in a different database than the one being audited.

  A new parameter, Database_File=db-name, has been added to the Audit qualifier. This parameter allows you to load the security audit records for one database into another database. ♦

– Loading .unl files created under another version of Oracle Rdb

  You can now unload a table from a database structured under one version of Oracle Rdb and load it into the same table of a database structured under another version of Oracle Rdb. For example, if you unload the EMPLOYEES table from a mf_personnel database created under Oracle Rdb V6.0, you can load the generated .unl file into an Oracle Rdb V7.0 database. Likewise, if you unload the EMPLOYEES table from a mf_personnel database created under Oracle Rdb V7.0, you can load the generated .unl file into an Oracle Rdb V6.1 database. This is true even for specially formatted binary files (created with the RMU Unload command without the Record_Definition qualifier). The earliest version into which you can load a .unl file from another version is Oracle Rdb V6.0.

  See the *Oracle RMU Reference Manual* for details.

– Storing null values

  A new option, Null, has been added to the Record_Definition qualifier. This qualifier allows you to store null values.

See the *Oracle RMU Reference Manual* and the *Oracle Rdb7 Guide to Database Design and Definition* for details on all these new qualifiers, parameters, and options.

- New RMU Load Plan command

  The RMU Load Plan command allows you to execute a load plan file created with the new List_Plan qualifier of the RMU Load command.

  See the *Oracle RMU Reference Manual* and the *Oracle Rdb7 Guide to Database Design and Definition* for details.

- RMU Move_Area command enhancement

  A new qualifier, Transaction_Mode, sets the allowable transaction modes for the database root file created by this command. For more information, see Section 4.4.1.

- RMU Optimize After_Journal command enhancement

  The RMU Optimize After_Journal command now allows you to specify a qualifier that directs Oracle RMU to keep the label it finds on a tape during an .aij optimization operation even if that label does not match the default label or that specified with the Label qualifier. The new qualifier is the Accept_Labels qualifier.

  See the *Oracle RMU Reference Manual* for details.

- RMU Recover command enhancement

  The RMU Recover command now provides the Automatic and Noautomatic qualifiers. These qualifiers allow you to specify whether or not Oracle RMU should attempt automatic recovery of all .aij files.

  See the *Oracle RMU Reference Manual* for details.

- RMU Repair command enhancement

  The RMU Repair command now provides the Checksum qualifier. This qualifier allows you to check and update the checksums for all or specified areas in the database.

  See the *Oracle RMU Reference Manual* for details.

- RMU Restore command changes

  - Just_Page qualifier replaced

    The Just_Page qualifier is no longer valid for the RMU Restore and Recover commands. The Just_Page qualifier is replaced by the Just_Corrupt qualifier. See Section 1.10.2 for more information.

  - Transaction_Mode qualifier

    A new qualifier, Transaction_Mode, sets the allowable transaction modes for the database root file created by this command. For more information, see Section 4.4.1.

  - RMU Restore Only_Root command enhancement

    A new qualifier, Transaction_Mode, sets the allowable transaction modes for the database root file created by this command. For more information, see Section 4.4.1.

- RMU Server After_Journal Start command enhancement

The RMU Server After_Journal Start command now provides the Output qualifier. This qualifier allows you to provide a file specification for the AIJ log server (ALS) output file. Use the new RMU Server After_Journal Reopen_Output command to view this log file.

See the *Oracle RMU Reference Manual* for details.

- New RMU Server After_Journal Reopen_Output command

  This command allows you to view the current output file for the ALS.

  See the *Oracle RMU Reference Manual* for details.

- New RMU Server Backup_Journal Resume command

  This command reinstates after-image journal backup operations that were suspended with the RMU Server Backup_Journal Suspend command.

  See the *Oracle RMU Reference Manual* for details.

- New RMU Server Backup_Journal Suspend command

  This command allows you to temporarily suspend .aij backup operations on all database nodes. While operations are suspended, you cannot back up .aij files manually (with the RMU Backup After_Journal command) nor will the AIJ backup server (ABS) perform .aij backup operations.

  See the *Oracle RMU Reference Manual* for details.

- RMU Set After_Journal command enhancement

  The RMU Set After_Journal command now provides an Edit_Filename keyword option to the Add qualifier. This keyword allows you to specify a default edit string for the backup of a specific .aij file. Use the Backups qualifier with the Edit_Filename keyword to specify a default edit string for all .aij backup files.

  See the *Oracle RMU Reference Manual* for details.

- RMU Set Corrupt_Pages command enhancement

  The RMU Set Corrupt_Pages command now allows you to specify snapshot areas with the Areas qualifier and pages in the snapshot area with the Page qualifier.

  See the *Oracle RMU Reference Manual* for details.

- RMU Show After_Journal command enhancements

  The following new features are provided by the RMU Show After_Journal command:

  – The new Edit_Filename option that you can specify with the RMU Set After_Journal command is displayed by the RMU Show After_Journal command.

  – The full file specification of an added .aij file is displayed as a comment field. Previously, if the user did not enter a full file specification when he or she created the .aij file, a full file specification was not displayed when the user issued the RMU Show After_Journal command.

OpenVMS VAX OpenVMS Alpha

- Two new global process symbols, RDM$AIJ_ENDOFFILE and RDM$AIJ_FULLNESS, are defined when you specify the Backup_Context qualifier with the RMU Show After_Journal command. RDM$AIJ_ENDOFFILE contains the end of file block number for the current .aij file. RDM$AIJ_FULLNESS contains the percent fullness of the current .aij file. ♦

See the *Oracle RMU Reference Manual* for details.

- RMU Show Statistics command enhancements

  Several new qualifiers have been added to the RMU Show Statistics command. For more information, see Section 1.11.

- RMU Show System and Show Users command enhancement

  The RMU Show Users and Show System commands have been enhanced to display the number of available monitor message buffers, the date and time of the monitor start, and whether after-image backup operations have been temporarily suspended (with the RMU Server Backup_Journal Suspend command).

  See the *Oracle RMU Reference Manual* for details.

- RMU Unload command enhancement

  A new option, Null, has been added to the Record_Definition qualifier. This option allows you to unload Null values as a string that you specify to identify the Null value.

  See the *Oracle RMU Reference Manual* and the *Oracle Rdb7 Guide to Database Design and Definition* for details.

- RMU Verify command enhancements

  The following enhancements have been made to the RMU Verify command:

  - Detected asynchronous prefetch enabled

    Detected asynchronous prefetch should be enabled to achieve the best performance of this command. Beginning with Oracle Rdb V7.0, by default, detected asynchronous prefetch is enabled.

  - Enhancements to verification of pointers to table rows

    The RMU Verify command has changed the way it verifies pointers to table rows in indexes. In prior versions, RMU Verify tried to retrieve the table row pointed to by an index. Now, the Oracle RMU verify operation creates a sorted list of all dbkeys for a table and a sorted list of all dbkeys in an index and, displays discrepancies it finds between the two lists. This helps the verify operation run faster and, it allows the operation to detect any cases of an index missing an entry for a data row.

  - Enhancements to the Constraints qualifier

    The Constraints qualifier now lets you specify one or more specific constraints or tables for which constraints should be verified.

  - Routine verification provided

    The Routine qualifier lets you specify that all routines (functions and procedures) stored in the database are to be verified by Oracle RMU. The qualifier is synonymous with the Functions qualifier.

  - Logical area verification included with index verification

When you specify the Index and Data qualifiers with the RMU Verify command, Oracle RMU now includes verification of all the logical areas referenced by the specified index or indexes.

See the *Oracle RMU Reference Manual* for details.

- Windows Statistics command is no longer valid

  The RMU Windows Statistics command is obsolete for Oracle Rdb V7.0. See Section 1.5.3 for details.

- RMUwin DECwindows Motif interface is obsolete

  The RMUwin DECwindows Motif interface is no longer available on OpenVMS or Digital UNIX. See Section 1.5.3 for details.

1.10.1 Enhancements to the RMU Convert Command

In Oracle Rdb V7.0, index prefix cardinalities are automatically maintained. Databases prior to V7.0 do not contain index prefix cardinality information. Therefore, when a database is converted to V7.0 or higher, the RMU Convert command collects and stores the index prefix cardinalities for each multisegmented sorted index in the database.

Because the collection of index prefix cardinality information upon conversion of a large database can take a considerable amount of time, the RMU Convert command issues a new prompt that gives you the option of either computing or estimating the index prefix cardinalities. The default is to estimate. If you opt for estimation, the index prefix cardinalities are derived from the existing index cardinality stored in the database.

If you choose the estimate option, Oracle Rdb highly recommends that you execute an RMU Collect Optimizer_Statistics command after successful database conversion. Doing so replaces the estimated index prefix cardinality values with the real values. The following example shows the command syntax for OpenVMS:

```
$ RMU/COLLECT OPTIMIZER_STATISTICS <root-file-spec>/NOTABLES -
$_ /STATISTICS=CARDINALITY
```

Because of the NoTables qualifier, this command collects only index cardinalities including index prefix cardinalities.

1.10.2 Just_Page Qualifier for RMU Restore and Recover Replaced with Just_Corrupt Qualifier

The Just_Page qualifier is no longer valid for the RMU Restore and Recover commands. The Just_Page qualifier is replaced by the Just_Corrupt qualifier. The Just_Corrupt qualifier differs from the Just_Page qualifier in the following ways:

- In previous versions of Oracle Rdb, if you specified the Just_Page qualifier as a local qualifier for an area, you could set individual pages in an area corrupt at the beginning a restore operation. The Just_Corrupt qualifier does not allow you to set specific pages corrupt. If you need to set specific pages corrupt, use the RMU Set Corrupt_Pages command before executing the RMU Restore command. By disallowing additions to the corrupt page table as part of the restore operation, Oracle RMU prevents some unusual situations from occurring (for example, overflowing the corrupt page table while adding corrupt pages during a restore operation).

```
$ ! In V6.0 and 6.1, you could add corrupt pages to the
$ ! corrupt page table as part of the restore operation:
$ !
$ RMU/RESTORE MF_PERSONNEL.RBF EMPIDS_LOW/JUST_PAGE=60
$ !
$ ! Beginning with V7.0, setting corrupt pages is
$ ! separate from restoring corrupt pages:
$ !
$ RMU/SET CORRUPT_PAGES /CORRUPT /AREA=EMPIDS_LOW /PAGE=60 MF_PERSONNEL
$ RMU/RESTORE MF_PERSONNEL.RBF EMPIDS_LOW/JUST_CORRUPT
```

- Prior to V7.0, if you specified the Just_Page qualifier as a global qualifier for the RMU Restore command, Oracle RMU would try to restore all corrupt pages in the corrupt page table. Now, if you specify the Just_Corrupt qualifier as a global qualifier for the RMU Restore command, Oracle RMU restores all corrupt storage areas and corrupt pages. This allows you to specify one Restore command and allows Oracle RMU to make one pass through the backup file to restore all known corruptions in the database.

```
$ ! In V6.0 and 6.1, you restored corrupt areas with one RMU Restore command,
$ ! and restored corrupt pages with a second RMU Restore command:
$ !
$ RMU/RESTORE/JUST_PAGES MF_PERSONNEL.RBF
$ RMU/RESTORE/AREA MF_PERSONNEL.RBF <LIST OF CORRUPT AREAS>
$ !
$ ! Beginning with V7.0, you restore corrupt areas and corrupt
$ ! pages with one RMU Restore command:
$ !
$ RMU/RESTORE/JUST_CORRUPT MF_PERSONNEL.RBF
```

- Previous versions did not let you mix a by-area restore operation of some areas with a just-page restore of other areas in the same command. Now, you can now combine these two operations in the same command. Note that if you specify the Recover qualifier with the Restore command, the recover operation recovers all areas and pages that were restored during the restore operation.

```
$ ! In V6.0 and 6.1, you were required to use separate commands for restoring
$ ! areas and restoring corrupt pages:
$ !
$ RMU/RESTORE/AREA MF_PERSONNEL.RBF EMPIDS_MID
$ RMU/RESTORE/AREA MF_PERSONNEL.RBF EMPIDS_LOW, JOBS
$ !
$ ! Beginning with V7.0, you can combine a by-area restore
$ ! operation and a restore of corrupt pages in one command:
$ !
$ RMU/RESTORE MF_PERSONNEL.RBF EMPIDS_LOW, EMPIDS_MID/JUST_CORRUPT, JOBS
```

- In previous versions, if you specified the Just_Page qualifier with the Recover command, Oracle RMU recovered only inconsistent pages. Now, when you specify the Just_Corrupt qualifier with the Recover command, Oracle RMU recovers all inconsistent areas and inconsistent pages The behavior of automatic recovery has been changed so that Oracle RMU recovers inconsistent pages as well as inconsistent areas. This makes the automatic recovery operation semantically the same as an RMU Recover command with the Just_Corrupt qualifier specified. Request an automatic recover operation by specifying the Area qualifier without a list of areas.

```
$ ! In V6.0 and 6.1, you specified recovery of inconsistent areas in one
$ ! command and recovery of inconsistent pages in another command:
$ !
$ RMU/RECOVER/AREA MF_PERSONNAL.AIJ EMPIDS_LOW, JOBS
$ RMU/RECOVER /JUST_PAGES MF_ERSONNEL.AIJ
$ !
$ ! Beginning with V7.0, you can specify recovery of inconsistent areas and
$ ! pages in one command. You can use either of the following two commands
$ ! to obtain the same results:
$ !
$ RMU/RECOVER/AREA MF_PERSONNEL.AIJ
$ !
$ ! or:
$ !
$ RMU/RECOVER/JUST_CORRUPT MF_PERSONNEL.AIJ
```

1.11 New Features in the Performance Monitor

The following sections describe new features for the Performance Monitor (the RMU Show Statistics utility.)

In addition to the new screens and features described in the following sections, the Performance Monitor contains the following new features:

- Stall Message logging

  The RMU Show Statistics command has been enhanced to allow you to write all stall messages to a log.

  The Stall_Log=file-spec qualifier specifies that stall messages are to be written to the specified file. This can be useful when you notice a great number of stall messages being generated, but do not have the resources on hand to immediately investigate and resolve the problem. The file generated by the Stall_Log qualifier can be reviewed later so that the problem can be traced and resolved. You can enable stall message logging at any time. For more information, see the *Oracle RMU Reference Manual* and the *Oracle Rdb7 Guide to Database Performance and Tuning*.

- Reopening output files

  You can now access the database statistics output file without having to detach from the database and rundown the image. A new RMU Show Statistics qualifier, Reopen_Interval=minutes, causes the utility to periodically close the current output file and open a new file.

  After the specified interval, it closes the current output file and opens a new output file without requiring you to exit from the Performance Monitor.

  This qualifier allows you to view data written to the output file while the Performance Monitor is running. For more information, see the *Oracle RMU Reference Manual*.

- Dbkey logging

  Tools like the Performance Monitor can help solve a current problem within the database. But most problems are solved within a limited timeframe. Because most customers are running Oracle Rdb in a 7x24 hour environment, it is not always possible to have an expert readily available. This means most customers cannot trace a problem until after it has occurred and been solved.

The Performance Monitor now provides a dbkey logging mechanism, the RMU Show Statistics Dbkey_Log=file-spec qualifier, to record the dbkeys and help identify the records accessed during a given processing period by the various attached processes. This log can help an expert to trace a problem. You can enable logging at any time.

For more information, see the *Oracle RMU Reference Manual*.

- Database Dashboard facility

  The Database Dashboard displays the actual database parameter and attribute settings used by the processes attached to the database, both the global database and individual database processes. Optionally, it allows the dynamic and non-persistent updating of certain database parameters and attributes on a single node at run time. You can actively or passively broadcast to all active database processes on the node upon which the changes were made. You can examine the net effect of these changes at run time without having to restart database processes.

  The facility is known as the Database Dashboard because you can actually “drive” the database faster or slower and immediately see the impact or benefit of changing certain database settings higher or lower.

  For more information, see the Performance Monitor Help.

- Easier page migration

  When viewing a screen containing multiple pages, you now can use the up-arrow and down-arrow keys to migrate around the various pages in a circular manner. The ">" and "<" continue to operate as before; that is, the ">" stops at the last page and the "<" stops at the first page.

  For instance, when positioned on the first page of a screen, the "<" key does not change pages, while the up-arrow key moves to the last page of the screen. Conversely, when positioned on the last page of a screen, the ">" key does not change pages, while the down-arrow key moves to the first page of the screen.

  For more information, see the *Oracle Rdb7 Guide to Database Performance and Tuning*.

- Automatically migrating through the Performance Monitor screens

  The Performance Monitor now includes a new qualifier, [No]Cycle=seconds, that lets you automatically cycle through the set of screens associated with the currently selected menu item. When you specify the Cycle qualifier, you can change screen modes or change submenus as desired; cycling through the menus associated with your choice at whichever menu level is currently selected.

  When the Cycle qualifier is specified, the Performance Monitor continues to operate normally. You can change screen modes or change submenus at will, however, the cycle operation continues at the current menu level.

  For more information, see the *Oracle Rdb7 Guide to Database Performance and Tuning* and the *Oracle RMU Reference Manual*.

- Search mode for Performance Monitor screens

  Often, it is necessary to quickly locate a Performance Monitor screen that contains activity, because it is not always apparent what database activity is occurring. Previously, the only method available to accomplish this was to manually migrate through the available statistics screen and search the screen for activity.

You can now use the space bar to initiate a search for the next data screen in the current submenu group that has current activity. If there is no screen in the current submenu group that has activity, the Performance Monitor places you at the next screen, exactly as if you had used the Next Screen key. Also, computational screens (such as the Row Cache Computations screen) and informational screens (such as the Stall Messages, Monitor Log or AIJ Journal Information screens) are ignored during the search for active data.

The search-mode is available during replay of a binary input file.

For more information, see the *Oracle Rdb7 Guide to Database Performance and Tuning*.

- Zoom screens

  It is often necessary to quickly review detailed information about a storage area or active database process. Typically, this need occurs while reviewing one of the by-area or per-process screens when some event occurs that warrants further investigation.

  Zoom screens have been added to most by-area and per-process screens. A **zoom screen** is a subwindow that displays detailed information about a specific screen item, typically a lock, process or storage area. Note that a zoom screen is a static snapshot of information. Unlike normal statistics displays, the screen information does not change.

  A screen that has zoom capability displays the zoom onscreen-menu option. The following screens have zoom screen capability:

  Stall Messages
  Active Stall Messages
  Process Accounting
  Checkpoint Information
  CPU Utilization
  DBR Activity
  Lock Timeout History
  Lock Deadlock History
  DBKEY Information
  File IO Overview
  File Locking Overview
  AIJ Journal Information

- Pausing the Performance Monitor display

  The Pause onscreen menu option causes the screen to pause the output without affecting the operation of the utility.

  Press the P key to pause the screen display. The Pause onscreen menu option is highlighted, indicating that the display is paused. While the display is paused, all onscreen menu options are operational, therefore, broadcast messages continue to be delivered. To release the screen display, press the P key again.

  Only screens that have the pause onscreen-menu option are affected by the pause screen. Other screens continue to be updated and cannot be paused. Note that while a screen is paused you are free to migrate to other screens without affecting the pause status.

- Switching databases without exiting the Performance Monitor

You can switch databases without exiting the Performance Monitor by using the new menu option, Switch Database, of the Notepad Facility. When you select this option, the Performance Monitor prompts you for the file specification for the new database you want to open. When you enter a new database file specification, the current database is closed and the new database is opened. If the new database cannot open for any reason, the previously opened database is opened again.

The ability to open a new database is not available if you are replaying a binary input file, using the Input qualifier, or recording a binary output file, using the Output qualifier.

- Moving more easily between screens using plus (+) and minus (-) keys

  You can use the plus sign (+) to move forward and the minus sign (-) to move backward through screens. When you enter either the plus sign (+) or minus sign (-), the utility prompts you to enter the number of screens to move forward or backward. However, if you enter the number 0, the utility accesses the first or last screen depending on the keystroke command:

  - When using the plus sign (+), the utility moves to the last screen
  - When using the minus sign (-) , the utility moves to the first screen

OpenVMS VAX OpenVMS Alpha

- Displaying elapsed number of days

  The Performance Monitor now displays the number of elapsed days if the elapsed time has exceeded 24 hours. ♦

- Changes to the onscreen menu options

  The Performance Monitor has expanded onscreen menu options, causing the following changes for Version 7.0:

  - Display_menu is now Menu. You can now use either the D key (as before) or the M key (new) to bring up the main menu.
  - Write_screen is now Write.

1.11.1 New Screens

The Performance Monitor now contains the following new screens and menus:

- Lock Statistics submenu

  The Lock Statistics submenu contains a class of screens that displays information about locks that are specific to database storage areas and snapshot areas. This information is vital in determining which storage areas have the most locking activity and analyzing the validity of storage area partitioning.

  The Lock Statistics submenu is accessed from the main menu.

- Online Analysis facility

  The Online Analysis facility identifies items of interest to the DBA for further investigation into possible performance problems. It contains nine analysis screens. The Online Analysis facility is available only during online database access. It is not available during replay of a binary input file.

  The Online Analysis facility is accessed from the main menu.

- Recovery Statistics screen

The Recovery Statistics screen provides information useful to the DBA in determining proper database attributes and parameter settings that ensure timely and efficient process recovery. It identifies various recovery phases and collects information on how long it takes each phase to complete. The screen provides global information on all failed process recoveries, not individual process recoveries.

The Recovery Statistics screen is accessed from the Journal Information submenu.

- 2PC Statistics screen

  The 2PC Statistics screen provides information on the performance of distributed (2PC) transactions. It provides information about how distributed transaction performance differs from non-distributed transaction performance. The information displayed on the screen is written to the binary output file and is available during binary input file replay.

  The 2PC Statistics screen is accessed from the Journal Information submenu.

- Active User Chart

  The Active User Chart graphically portrays the number of currently active users attached to the database over a measured period of time.

  This screen is accessed from the Process Information submenu.

- DBKEY Information screen

  The DBKEY Information screen allows you to determine which database pages are being accessed by any or all processes attached to the database. The screen is especially useful for identifying collisions on hot pages.

  This screen is accessed from the Process Information submenu.

- Monitor Log screen

  The Monitor Log screen allows you to view the monitor log online, even when disk-based logging is disabled because of disk-space problems. Also, it allows you to know when the monitor logging is disabled.

  This screen is accessed from the Process Information submenu.

- AIJ Journal Growth Trend screen

  The AIJ Journal Growth Trend screen graphically portrays the size of the current AIJ Journal over a measured period of time.

  This screen is accessed from the Journaling Information submenu.

- ALS Statistics screen

  The ALS Statistics screen allows you to track the effectiveness of the AIJ Log Server process.

  This screen is accessed from the Journaling Information submenu.

- RUJ Statistics screen

  The RUJ Statistics screen contains summary information for all active update transactions on the current node.

  This screen is accessed from the Journaling Information submenu.

OpenVMS VAX OpenVMS Alpha

- OpenVMS SYSGEN Parameters screen

  The SYSGEN Parameters screen allows you to review SYSGEN parameter settings without exiting from the Performance Monitor.

  This screen is accessed from the Database Parameter Information submenu. ♦

- RCS Statistics screen

  The RCS Statistics screen provides information on the run-time operation of the Row Cache Server process.

  This screen is accessed from the Row Cache Information submenu.

- Row Cache Queue Length screen

  While the row cache feature saves I/O operations, poor CPU performance can result if many hash table collisions occur. The Row Cache Queue Length screen helps determine the relative CPU performance impact of the caching of the rows.

  This screen is accessed from the Row Cache Information submenu.

- Row Cache Length screen

  This screen graphically describes the distribution of the various row lengths within a particular row cache.

  To know how well sized the row cache is, you must know the distribution of the various row lengths in the row cache. For instance, knowing that each cache entry is wasting even 5 bytes is significant if there are 100,000 entries in the cache.

  This screen is accessed from the Row Cache Information submenu.

- Summary Cache Statistics screen

  The Summary Cache Statistics screen provides summary information for all row caches in the database.

  This screen is accessed from the main menu.

- Row Cache (One Cache) screen

  The Row Cache (One Cache) screen provides summary information for a specific row cache. This screen is similar to the Summary Cache Statistics screen, except that it displays the data for a particular row cache.

  This screen is accessed from the main menu.

- Row Cache Utilization screen

  The Row Cache Utilization screen provides utilization information in a graphical format for a specific row cache.

  This screen is accessed from the Row Cache Information submenu.

- Hot Record Information screen

  Hot Record Information screen identifies the most frequently accessed records for a specific row cache.

  This screen is accessed from the Row Cache Information submenu.

- Row Cache Status screen

  The Row Cache Status screen provides overall status for a specific row cache.

  This screen is accessed from the Row Cache Information submenu.

OpenVMS VAX OpenVMS Alpha

- Device Information screen

  The Device Information screen provides an online view of the storage-area device information local to a particular database.

  This screen is accessed from the IO Statistics (by file) submenu. ♦

- Scatter Plot screen

  You can now determine the composition of average rate statistic values using a new class of screen display known as the Scatter Plot. The Scatter Plot display looks similar to the Transaction Duration display. However, it shows information for a selected statistics field on the current screen, similar in many respects to the existing Time Plot display. The Scatter Plot display shows information on the current rate of that statistics field in the form of a vertical histogram.

  You select the Scatter Plot display with the X_plot of the onscreen menu option. After entering the X, a menu of the existing screen fields is displayed. Then, you select the single statistics field that you want to examine.

- Database Parameter Information submenu

  The Database Parameter Information submenu allows you to examine database parameter settings from within the Performance Monitor.

  The Database Parameter Information submenu has the following options:

  A. General Information

  B. Buffer Information

  C. Lock Information

  D. Storage Area Information

  E. Row Cache Information

  F. Journaling Information

  G. Fast Commit Information

  H. Hot Standby Information

  I. Audit Information

  J. Active User Information

  K. OpenVMS SYSGEN Information (OpenVMS only)

  The Database Parameter Information submenu is accessed from the main menu.

For more information about the new screens, see the Performance Monitor Help.

1.11.2 Enhancements to Existing Screens

The Performance Monitor provides the following enhancements to existing screens:

- Per-Area Pages Checked Statistics

  Oracle Rdb collects pages checked statistics on a per file basis and displays the information in the File IO Overview screen.

- Filter onscreen-menu option for the Stall Messages screen

The Filter option allows you to enter a case-insensitive search string that is used to filter the stall messages. Only those stall messages that contain the specified search string are displayed. The Filter option is also available using the Config onscreen-menu option used to configure the screen display.

For more information, see the *Oracle Rdb7 Guide to Database Performance and Tuning* and the Performance Monitor help.

- Checkpoint Information screen

  The Checkpoint Information screen has a new Config onscreen-menu option that contains the following configuration options:

  A. Display Transaction Elapsed Time

  B. Unsorted Display

  C. Sort by oldest active checkpoint

  D. Sort by oldest active transaction

  E. Sort by oldest quiet point

  For more information, see the Performance Monitor help.

- Sorting in File IO Overview screen

  The Performance Monitor on Windows NT, Windows95, and Windows 3.1 now allows sorting of the File IO Overview screen. To sort on a given column, click on the column header.

  Note that there is a known problem in that reconfigured screens are not sorted properly. If you add or delete columns from the display screen, subsequent sorts may not be correct.

1.12 System Metadata Changes

Oracle Rdb V7.0 has added or changed the following system metadata components:

- New domain definitions
- New creator columns added to record the user name of the creator of the object.
- New optional system table, RDB$WORKLOAD
- New generic columns, RDB$CREATED and RDB$LAST_ALTERED, added to record timestamps of metadata changes and a new column for RDB$FIELDS table to be used as a bitmask
- New index for the Replication Option for Rdb optional system table RDB$TRANSFER_RELATIONS
- New LIST OF BYTE VARYING output for system tables

1.12.1 Domain Changes

A new system domain, RDB$CARDINALITY, is now used for the RDB$CARDINALITY column in RDB$INDICES and RDB$RELATIONS. It is also used for the new RDB$CARDINALITY column in RDB$INDEX_SEGMENTS.

A new system domain, RDB$FACTOR, is used for several new columns which record fractional, or approximate values.

A new system domain, RDB$PROBABILITY, is used for the RDB$NULL_FACTOR column to record the fraction of rows which contain NULL.

A new system domain, RDB$TIMESTAMP, is used for all the RDB$LAST_ALTERED and RDB$CREATED columns.

A new system domain, RDB$TINY_BIT_MASK, is used for the new RDB$FLAGS column for the RDB$FIELDS table.

1.12.2 Owner and Create/Alter Timestamps

Two timestamp columns and a creator column have been added to the following tables, which describe created objects:

RDB$CATALOG_SCHEMA (optional multischema table)
RDB$COLLATIONS
RDB$CONSTRAINTS
RDB$DATABASE
RDB$FIELDS
RDB$INDICES
RDB$MODULES
RDB$QUERY_OUTLINES
RDB$RELATIONS
RDB$ROUTINES
RDB$TRIGGERS
RDB$WORKLOAD (optional workload collection table)

The timestamps record when the object was created (or converted from a version prior to V7.0), and when it was last altered by the user. Note that updates due to cardinality changes or index root dbkeys are not recorded. This change is based on a recent change to the SQL PSM (Persistent Stored Modules) definition. The PSM is likely to become an addendum to SQL92, the current ANSI and ISO SQL Standard.

The object creator column records the current user name of the user who created the object (or converted the database from a version prior to V7.0).

1.12.3 Optional RDB$WORKLOAD Table

RDB$WORKLOAD is an optional system table, similar to RDB$SYNONYMS and RDB$CATALOG_SCHEMA. It is created when you specify the WORKLOAD COLLECTION IS ENABLED clause in a CREATE or ALTER DATABASE statement. Once created, this table can never be dropped. Optional system tables also have row compression enabled to conserve space.

A unique index, RDB$WRKLD_ID_FLD_NDX is also created on the two columns (RDB$RELATION_ID, RDB$FIELD_GROUP).

Table 1–2 shows the columns for the RDB$WORKLOAD Table.

Table 1–2 Columns for RDB$WORKLOAD Table

Column Name	Data Type	Domain Name	Comments
RDB$CREATED	DATE VMS	RDB$TIMESTAMP	Time profile entry was created
RDB$LAST_ALTERED	DATE VMS	RDB$TIMESTAMP	Last time statistics were updated

(continued on next page)

Table 1–2 (Cont.) Columns for RDB$WORKLOAD Table

Column Name	Data Type	Domain Name	Comments
RDB$DUPLICITY_FACTOR	BIGINT(7)	RDB$FACTOR	Value ranges from 1.0 to table cardinality. Number of duplicate values for an interesting column group (RDB$FIELD_GROUP). The pathological case is when all rows have the same value for an interesting column group, making the duplicity factor equal to table cardinality. (This extreme case should rarely happen.)
RDB$NULL_FACTOR	INTEGER(7)	RDB$PROBABILITY	Value ranges from 0.0 to 1.0. This is the proportion of table rows that have NULL in one or more columns of an interesting column group
RDB$RELATION_ID	INTEGER	RDB$OBJECT_ID	Base table ID
RDB$FLAGS	INTEGER	RDB$BIT_MASK	Reserved for future use
RDB$FIELD_GROUP	CHAR(31)	RDB$OBJECT_NAME	Contains up to 15 sorted field-ids
RDB$SECURITY_CLASS	CHAR(20)	RDB$SECURITY_ CLASS	Reserved for future use

1.12.4 Modified System Tables

The following tables describe additional or modified columns for Oracle Rdb V7.0.

Table 1–3 shows the new columns for the RDB$COLLATIONS table.

Table 1–3 New Columns for RDB$COLLATIONS Table

Column Name	Data Type	Domain Name	Comments
RDB$CREATED	DATE VMS	RDB$TIMESTAMP	Set when the collating sequence is created
RDB$LAST_ALTERED	DATE VMS	RDB$TIMESTAMP	Reserved for future use
RDB$COLLATION_CREATOR	CHAR(31)	RDB$OBJECT_OWNER	Creator of this collating sequence

Table 1–4 shows the new columns for the RDB$CONSTRAINTS table.

Table 1–4 New Columns for RDB$CONSTRAINTS Table

Column Name	Data Type	Domain Name	Comments
RDB$CREATED	DATE VMS	RDB$TIMESTAMP	Set when the constraint is created
RDB$LAST_ALTERED	DATE VMS	RDB$TIMESTAMP	Reserved for future use
RDB$CONSTRAINT_CREATOR	CHAR(31)	RDB$OBJECT_OWNER	Creator of this constraint

Table 1–5 shows the new columns for the RDB$DATABASE table.

Table 1–5 New Columns for RDB$DATABASE Table

Column Name	Data Type	Domain Name	Comments
RDB$CREATED	DATE VMS	RDB$TIMESTAMP	Set when the database is created

(continued on next page)

Table 1–5 (Cont.) New Columns for RDB$DATABASE Table

Column Name	Data Type	Domain Name	Comments
RDB$LAST_ALTERED	DATE VMS	RDB$TIMESTAMP	Set when you use ALTER DATABASE to change the RDB$DATABASE row
RDB$DATABASE_CREATOR	CHAR(31)	RDB$OBJECT_OWNER	Creator of this database
RDB$DEFAULT_STORAGE_ AREA_ID	INTEGER	RDB$OBJECT_ID	Default storage area used for unmapped persistent tables and indexes
RDB$DEFAULT_TEMPLATE_ AREA_ID	INTEGER	RDB$OBJECT_ID	Reserved for future use

Table 1–6 shows the new columns for the RDB$FIELDS table.

Table 1–6 New Columns for RDB$FIELDS Table

Column Name	Data Type	Domain Name	Comments
RDB$FLAGS	TINYINT	RDB$TINY_BIT_MASK	
RDB$CREATED	DATE VMS	RDB$TIMESTAMP	Set when the domain is created
RDB$LAST_ALTERED	DATE VMS	RDB$TIMESTAMP	Set when ALTER DOMAIN is used
RDB$FIELD_CREATOR	CHAR(31)	RDB$OBJECT_OWNER	Creator of this domain

In a future version, the RDB$FLAGS column will be changed to an INTEGER data type to be consistent with the existing RDB$FLAGS columns in other tables. The new domain RDB$TINY_BIT_MASK is created only to support the data type of the new RDB$FLAGS column for RDB$FIELDS table, and may not exist in a future version.

Table 1–7 shows the new columns for the RDB$INDEX_SEGMENTS table.

Table 1–7 New Columns for RDB$INDEX_SEGMENTS Table

Column Name	Data Type	Domain Name	Comments
RDB$CARDINALITY	BIGINT	RDB$CARDINALITY	Prefix cardinality for this and all prior key segments (assumes sorting by ordinal position)

Table 1–8 shows the new columns for the RDB$INDICES table.

Table 1–8 New Columns for RDB$INDICES Table

Column Name	Data Type	Domain Name	Comments
RDB$CREATED	DATE VMS	RDB$TIMESTAMP	Set when the index is created.
RDB$LAST_ALTERED	DATE VMS	RDB$TIMESTAMP	Set when ALTER INDEX is used.
RDB$INDEX_ CREATOR	CHAR(31)	RDB$OBJECT_ OWNER	Creator of this index.

(continued on next page)

Table 1–8 (Cont.) New Columns for RDB$INDICES Table

Column Name	Data Type	Domain Name	Comments
RDB$KEY_CLUSTER_ FACTOR	BIGINT(7)	RDB$FACTOR	Sorted Index: The ratio of the number of clump changes that occur when you traverse Level 1 index nodes and the duplicate node chains to the number of keys in the index. This statistic is based on an entire index traversal. This means the last duplicate node of the current key is compared with the first duplicate node of the next key for clump change.
			Hash Index: The average number of clump changes that occur when you go from system record to hash bucket to overflow hash bucket (if fragmented), and traverse the duplicate node chain for each key. This statistics is based on per-key traversal.
RDB$DATA_CLUSTER_ FACTOR	BIGINT(7)	RDB$FACTOR	Sorted Index: The ratio of the number of clump changes that occur between adjacent dbkeys in the duplicate chains of all keys to the number of keys in the index. For a unique index, the dbkeys of adjacent keys are compared for clump change. This statistic is based on an entire index traversal. This means the last dbkey of the current key is compared with the first dbkey of the next key for clump change.
			Hashed Index: The average number of clump changes that occur between adjacent dbkeys in a duplicate chain for each key. For a unique index, this value is always 1. This statistics is based on per-key traversal.
RDB$INDEX_DEPTH	INTEGER	RDB$COUNTER	Sorted Index: The depth of the B-tree.
			Hashed Index: This column is not used for hashed indexes and is left as 0.

Table 1–9 shows the changed columns for the RDB$INDICES table.

Table 1–9 Changed Columns for RDB$INDICES Table

Column Name	Data Type	Domain Name	Comments
RDB$CARDINALITY	BIGINT	RDB$CARDINALITY	Change to use new domain

Table 1–10 shows the new columns for the RDB$MODULES table.

Table 1–10 New Columns for RDB$MODULES Table

Column Name	Data Type	Domain Name	Comments
RDB$CREATED	DATE VMS	RDB$TIMESTAMP	Set when the module is created.
RDB$LAST_ALTERED	DATE VMS	RDB$TIMESTAMP	Set when ALTER MODULE is used (future).
RDB$MODULE_CREATOR	CHAR(31)	RDB$OBJECT_OWNER	Creator of this module. Differentiates between OWNER and AUTHORIZATION.

Table 1–11 shows the new columns for the RDB$QUERY_OUTLINES table.

Table 1–11 New Columns for RDB$QUERY_OUTLINES Table

Column Name	Data Type	Domain Name	Comments
RDB$CREATED	DATE VMS	RDB$TIMESTAMP	Set when the outline is created
RDB$LAST_ALTERED	DATE VMS	RDB$TIMESTAMP	Reserved for future use
RDB$OUTLINE_CREATOR	CHAR(31)	RDB$OBJECT_OWNER	Creator of this outline

Table 1–12 shows the new columns for the RDB$RELATIONS table.

Table 1–12 New Columns for RDB$RELATIONS Table

Column Name	Data Type	Domain Name	Comments
RDB$CREATED	DATE VMS	RDB$TIMESTAMP	Set when the table is created. (For system tables it is the same as the database creation timestamp.)
RDB$LAST_ALTERED	DATE VMS	RDB$TIMESTAMP	Set when ALTER TABLE, CREATE/ALTER STORAGE MAP, or ALTER DOMAIN cause changes to the table.
RDB$RELATION_CREATOR	CHAR(31)	RDB$OBJECT_OWNER	Creator of this table.
RDB$ROW_CLUSTER_ FACTOR	BIGINT(7)	RDB$FACTOR	The ratio of the number of clump changes that occur when you sequentially read the rows to the number of rows in a table. If a row is fragmented and part of its fragment is located in a clump different from the current one or immediate next one, it should be counted as a clump change.

Table 1–13 shows the changed columns for the RDB$RELATIONS table.

Table 1–13 Changed Columns for RDB$RELATIONS Table

Column Name	Data Type	Domain Name	Comments
RDB$CARDINALITY	BIGINT	RDB$CARDINALITY	Change to use new domain

Table 1–14 shows the new columns for the RDB$ROUTINES table.

Table 1–14 New Columns for RDB$ROUTINES Table

Column Name	Data Type	Domain Name	Comments
RDB$CREATED	DATE VMS	RDB$TIMESTAMP	Set when the routine is created. (The same as the parent module's creation timestamp.)
RDB$LAST_ALTERED	DATE VMS	RDB$TIMESTAMP	Reserved for future use.
RDB$ROUTINE_CREATOR	CHAR(31)	RDB$OBJECT_OWNER	Creator of this routine. Differentiates between AUTHORIZATION and OWNER.

Table 1–15 shows the new columns for the RDB$TRIGGERS table.

Table 1–15 New Columns for RDB$TRIGGERS Table

Column Name	Data Type	Domain Name	Comments
RDB$CREATED	DATE VMS	RDB$TIMESTAMP	Set when the trigger is created
RDB$LAST_ALTERED	DATE VMS	RDB$TIMESTAMP	Reserved for future use
RDB$TRIGGER_CREATOR	CHAR(31)	RDB$OBJECT_OWNER	Creator of this trigger

Table 1–15 shows the new columns for the RDB$CATALOG_SCHEMA table.

Table 1–16 New Columns for RDB$CATALOG_SCHEMA Table

Column Name	Data Type	Domain Name	Comments
RDB$SECURITY_CLASS	CHAR(20)	RDB$SECURITY_ CLASS	Reserved for future use
RDB$CREATED	DATE VMS	RDB$TIMESTAMP	Set when the schema or catalog is created
RDB$LAST_ALTERED	DATE VMS	RDB$TIMESTAMP	Reserved for future use
RDB$CATALOG_SCHEMA_ CREATOR	CHAR(31)	RDB$OBJECT_OWNER	Creator of this schema or catalog

1.12.5 Changes to RDB$TRANSFER_RELATIONS Table

The table RDB$TRANSFER_RELATIONS is created by the Replication Option for Rdb. In versions of Oracle Rdb prior to V7.0, it did not have an index associated with it and so Oracle Rdb was forced to perform sequential scans when loading metadata for a transfer enabled database.

To improve performance, Oracle Rdb now detects this table name during table creation and automatically creates the duplicates index RDB$TRAN_RELS_REL_ NAME_NDX on the column RDB$RELATION_NAME.

The duplicates index is also added when the database is converted with the RMU Convert command.

1.12.6 Metadata LIST OF BYTE VARYING Changes

In previous versions, Oracle Rdb has supported multiple segment LIST OF BYTE VARYING data types for user-defined data. However, Oracle Rdb maintained its own LIST OF BYTE VARYING metadata columns as single segments. This restricted the length to approximately 65530 bytes. A CREATE TRIGGER or CREATE MODULE statement could fail due to this restriction.

In V7.0, this limit has been lifted by changing the way Oracle Rdb stores its system metadata.

- For columns containing binary data, such as the binary representation of a query, routine, constraint, trigger action, computed by column, or query outline, Oracle Rdb breaks the data into pieces that best fit the system storage area page size. Thus, the segments are all the same size with a possible small trailing segment.

  The LIST OF BYTE VARYING column value is no longer fragmented, improving performance when reading system metadata.

- For columns containing text data, such as the SQL source (for elements such as triggers and views) and user-supplied comment strings, Oracle Rdb breaks the text at line boundaries (indicated by ASCII carriage returns and line feeds) and stores the text without the line separator. Thus, the segments are of varying size with a possible zero length for blank lines.

  An application can now easily display the LIST OF BYTE VARYING column value and the application no longer needs to break up the single text segment for printing.

No change is made to the LIST OF BYTE VARYING column values when a database is converted (using RMU Convert, RMU Restore, or SQL EXPORT /IMPORT) from a previous version.

Applications that read the Oracle Rdb system LIST OF BYTE VARYING column values must be changed to understand multiple segments. Applications that do not read these system columns should see no change to previous behavior. Tools such as the RMU Extract command and the SQL SHOW and EXPORT statements handle both the old and new formats of the system metadata.

1.13 Application Compatibility Between Oracle Rdb Versions

OpenVMS VAX OpenVMS Alpha

Versions of Oracle Rdb are upwardly compatible, with a few exceptions. Upwardly compatible means that you do not need to recompile or relink existing applications when you upgrade to a new version of Oracle Rdb.

All previous versions of Oracle Rdb are compatible with Version 7.0, except for the following releases:

- Oracle Rdb V4.0 and V4.0A

  Object files and executables are not upwardly compatible. Users must recompile and relink their applications to upgrade to a higher version.

  The incompatibilities are fixed in V4.0B.

- Oracle Rdb V4.2

  Executables are not upwardly compatible. However, the incompatibilities are fixed in the Mandatory Update V4.2A.

  To upgrade to a higher version, users must upgrade to V4.2A or relink their applications.

- Oracle Rdb for OpenVMS VAX V5.0 (This was a limited distribution release.)

  Executables are not upwardly compatible. Users must relink their applications to upgrade to a higher version.

1.14 Software Requirements

For information about the minimum versions of operating systems and software requirements for Oracle Rdb V7.0, see the web page at the following URL:

`http://www.oracle.com/products/servers/rdb/html/Matrix.html`

1.15 Documentation for This Release

The following sections list the documentation available for this release and the order numbers for each manual.

To order Oracle documentation, contact Oracle Support and Documentation Sales at 1-800-252-0303 or 415-506-5988.

1.15.1 Online Documentation Format

In previous releases, Oracle Rdb has provided online documentation in Bookreader format. For V7.0, we continue to provide this format. However, we will not provide Bookreader format in releases after V7.0. For the next release, we will provide a more portable format.

For V7.0, in addition to Bookreader format, we are providing some of the documentation in Adobe Acrobat format, HTML format, and Postscript files. The documentation in Bookreader, Postscript, and HTML format is available on the Oracle Rdb software CD–ROM. The documentation in Adobe Acrobat format is available on the Rdb Client kits CD–ROM.

Documentation in Adobe Acrobat Format

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

The documentation in Adobe Acrobat format is available on the Rdb Client kits CD–ROM, in the directory `RDBDOC`. The files are provided in a self-extracting archive (rdb7pdf.exe) file for Windows NT and Windows 95 systems.

To maintain the directory structure, use the `-d` qualifier when you run the archive file to extract the files.

The following table maps the book titles to the Acrobat format files:

Book Title	File Name
Before You Install Oracle Rdb7	N/A
Oracle Rdb7 Release Notes	N/A
Oracle Rdb7 Installation and Configuration Guide for OpenVMS	IGVMS
Oracle Rdb7 Installation and Configuration Guide for Digital UNIX	IGUNIX
Oracle Rdb7 Introduction to SQL	SQLINT
Oracle Rdb7 Guide to SQL Programming	GSP
Oracle Rdb7 SQL Reference Manual	SQLRM
Oracle Rdb7 Guide to Database Design and Definition	GDDD
Oracle Rdb7 Guide to Database Maintenance	GDM
Oracle Rdb7 Guide to Database Performance and Tuning	GDPT
Migrating Oracle Rdb7 Databases and Applications to OpenVMS Alpha	MIGAXP
Migrating Oracle Rdb7 Databases and Applications to Digital UNIX	MIGUNIX
Oracle RMU Reference Manual for OpenVMS	RMURM_V
Oracle RMU Reference Manual for Digital UNIX	RMURM_U
Oracle Rdb7 Guide to Distributed Transactions V6.1	DISTXN
Guide to Using the Oracle SQL/Services Client API	SQLCLNT
Oracle SQL/Services Server Configuration Guide	SQSCONF
Oracle SQL/Services Release Notes	N/A
Oracle SQL/Services Installation Guide for OpenVMS	SQSIG_V
Oracle SQL/Services Installation Guide for DigitalUNIX	SQSIG_U

Book Title	File Name
Oracle Rdb7 and Oracle CODASYL DBMS: Guide to Hot Standby Databases	HOTSTDBY

Documentation in HTML format

The documentation in HTML format is available on the Oracle Rdb CD–ROM in the directory `rdbdoc`. The HTML documentation is provided in three packages to make it easier to move the files to a location accessible by your browser:

- A saveset (rdb7html.a) for OpenVMS systems
- A tar (rdb7html.tar) file for Digital UNIX systems
- A self-extracting archive (rdb7html.exe) file for Windows NT and Windows 95 systems

  To maintain the directory structure, use the `-d` qualifier when you run the archive file to extract the files.

After you extract the files from the packages, see the introductory page `index.html` in the top-level directory for more information about the HTML documentation. Not all V7.0 documentation is available in HTML format.

1.15.2 Documentation for Oracle Rdb for OpenVMS

Table 1–17 lists the Oracle Rdb for OpenVMS documentation, including order numbers.

Table 1–17 Documentation for Oracle Rdb for OpenVMS

Set or Book Title	Order Number
Library Set (includes listed documentation)	A48507-1
Documents Created or Revised for Oracle Rdb for OpenVMS Release 7.0[1]	
Before You Install Oracle Rdb7 for OpenVMS	A42861-1
Oracle Rdb7 Release Notes	None
Oracle Rdb7 Installation and Configuration Guide for OpenVMS	A42342-1
Oracle Rdb7 Introduction to SQL	A40827-1
Oracle Rdb7 Guide to SQL Programming	A42867-1
Oracle Rdb7 SQL Reference Manual	A47579-1
Oracle Rdb7 Guide to Database Design and Definition	A41749-1
Oracle Rdb7 Guide to Database Maintenance	A41748-1
Oracle Rdb7 Guide to Database Performance and Tuning	A41747-1
Migrating Oracle Rdb7 Databases and Applications to OpenVMS Alpha	A40682-1
Oracle RMU Reference Manual for OpenVMS	A41741-1
Guide to Using the Oracle SQL/Services Client API	A41981-1
Oracle SQL/Services Server Configuration Guide	A41983-1
Oracle SQL/Services Release Notes	None

[1]Because documentation for the Hot Standby™ option is provided separately (with the Hot Standby license), it is not listed here.

(continued on next page)

Table 1–17 (Cont.) Documentation for Oracle Rdb for OpenVMS

Set or Book Title	Order Number
Documents Created or Revised for Oracle Rdb for OpenVMS Release 7.0[1]	
Oracle SQL/Services Installation Guide for OpenVMS	A42343-1
Documents Created or Revised for Oracle Rdb for OpenVMS Release 6.1 and Earlier	
Oracle Rdb7 Guide to Distributed Transactions	A40826-1

[1]Because documentation for the Hot Standby™ option is provided separately (with the Hot Standby license), it is not listed here.

Documentation for the Rdb Web Agent is available on the software media, in HTML format.

1.15.3 Documentation for Oracle Rdb for Digital UNIX

Table 1–18 lists the Oracle Rdb for Digital UNIX documentation, including order numbers.

Table 1–18 Documentation for Oracle Rdb for Digital UNIX

Set or Book Title	Order Number
Library Set (includes listed documentation)	A48656-1
Documents Created or Revised for Oracle Rdb for Digital UNIX Release 7.0[1]	
Before You Install Oracle Rdb7 for Digital UNIX	A47585-1
Oracle Rdb7 Release Notes	None
Oracle Rdb7 Installation and Configuration Guide for Digital UNIX	A41982-1
Oracle Rdb7 Introduction to SQL	A40827-1
Oracle Rdb7 Guide to SQL Programming	A42867-1
Oracle Rdb7 SQL Reference Manual	A47579-1
Oracle Rdb7 Guide to Database Design and Definition	A41749-1
Oracle Rdb7 Guide to Database Maintenance	A41748-1
Oracle Rdb7 Guide to Database Performance and Tuning	A41747-1
Migrating Oracle Rdb7 Databases and Applications to Digital UNIX	A40763-1
Oracle RMU Reference Manual for Digital UNIX	A41746-1
Guide to Using the Oracle SQL/Services Client API	A41981-1
Oracle SQL/Services Server Configuration Guide	A41983-1
Oracle SQL/Services Release Notes	None
Oracle SQL/Services Installation Guide for Digital UNIX	A48494-1
Documents Created or Revised for Oracle Rdb Release 6.1 and Earlier	
Oracle Rdb7 Guide to Distributed Transactions	A40826-1

[1]Because documentation for the Hot Standby option is provided separately (with the Hot Standby™ license), it is not listed here.

Documentation for the Rdb Web Agent is available on the software media, in HTML format.

2
Known Problems, Restrictions, and Other Notes

This chapter describes problems and restrictions relating to Oracle Rdb Version 7.0, and includes workarounds where appropriate. Unless otherwise noted, all notes apply to all platforms.

2.1 Known Problems and Restrictions in All Interfaces

This section describes known problems and restrictions that affect all interfaces for Version 7.0.

2.1.1 Reinstall V7.0 After Installing Previous Versions

Because of a change in the V7.0 Oracle Rdb and SQL image identifiers necessitated by the sale of Rdb from Digital Equipment Corporation to Oracle Corporation, installing a previous Oracle Rdb version on a system that already has V7.0 installed causes many SQL files and images to be incorrectly replaced. Because of this situation, you **must** reinstall V7.0 after you install any previous version, including any Mandatory Update (MUP) kit released before June 1995.

This situation might arise if you have a system with multiple versions of Oracle Rdb installed, one of which was V7.0, and you applied a MUP against one of the earlier versions.

Because ECOs of previous versions install correctly, reinstalling V7.0 after installing an ECO for a previous version ECO is not necessary.

2.1.2 Monitor ENQLM Minimum Increased to 32767

OpenVMS VAX OpenVMS Alpha

In previous versions, the Oracle Rdb monitor process (RDMMON) was created with a minimum lock limit (ENQLM) of 8192 locks. This minimum has been increased to 32767 locks (the OpenVMS maximum value). ♦

2.1.3 Hot Standby Database Option Does Not Support Replication on Digital UNIX

Digital UNIX

The ability to implement the Hot Standby software on a Digital UNIX system is unsupported at this time.

Although the *Oracle Rdb7 and Oracle CODASYL DBMS: Guide to Hot Standby Databases* documentation describes support on both the Digital UNIX and OpenVMS operating systems, you can implement Hot Standby databases for Oracle Rdb in OpenVMS environments only. ♦

2.1.4 Oracle Rdb Workload Collection Can Stop Hot Standby Replication

If you are replicating your Oracle Rdb database using the Oracle Hot Standby option, you must not use the workload collection option. By default, workload collection is disabled. However, if you enabled workload collection, you must disable it on the master database prior to performing a backup operation on that master database if it will be used to create the standby database for replication purposes. If you do not disable workload collection, it could write workload information to the standby database and prevent replication operations from occurring.

The workaround included at the end of this section describes how to disable workload collection on the master database and allow the Hot Standby software to propagate the change to the standby database automatically during replication operations.

Background Information

By default, workload collection and cardinality collection are automatically disabled when Hot Standby replication operations are occurring on the standby database. However, if replication stops (even for a brief network failure), Oracle Rdb potentially can start a read/write transaction on the standby database to write workload collection information. Then, because the standby database is no longer synchronized transactionally with the master database, replication operations cannot restart.

Note

The Oracle Rdb optimizer can update workload collection information in the RDB$WORKLOAD system table even though the standby database is opened exclusively for read-only queries. A read/write transaction is started during the disconnect from the standby database to flush the workload and cardinality statistics to the system tables.

If the standby database is modified, you receive the following messages when you try to restart Hot Standby replication operations:

```
%RDMS-F-DBMODIFIED, database has been modified; AIJ roll-forward not possible
%RMU-F-FATALRDB, Fatal error while accessing Oracle Rdb.
```

Workaround

To work around this problem, perform the following:

- On the master database, disable workload collection using the SQL clause WORKLOAD COLLECTION IS DISABLED on the ALTER DATABASE statement. For example:

  ```
  SQL> ALTER DATABASE FILE mf_personnel
  cont> WORKLOAD COLLECTION IS DISABLED;
  ```

  This change is propagated to the standby database automatically when you restore the standby database and restart replication operations. Note that, by default, the workload collection feature is disabled. You need to disable workload collection only if you previously enabled workload collection with the WORKLOAD COLLECTION IS ENABLED clause.

- On the standby database, include the Transaction_Mode qualifier on the RMU Restore command when you restore the standby database. You should set this qualifier to read-only to prevent modifications to the standby database when replication operations are not active. The following example shows the Transaction_Mode qualifier used in a typical RMU Restore command:

  ```
  $ RMU/RESTORE /TRANSACTION_MODE=READ_ONLY
                /NOCDD
                /NOLOG
                /ROOT=DISK1:[DIR]standby_personnel.rdb
                /AIJ_OPT=aij_opt.dat
                DISK1:[DIR]standby_personnel.rbf
  ```

If, in the future, you fail over processing to the standby database (so that the standby database becomes the master database), you can reenable updates to the “new” master database. For example, to reenable updates, use the SQL statement ALTER DATABASE and include the SET TRANSACTION MODES (ALL) clause. The following example shows this statement used on the new master database:

```
SQL> ALTER DATABASE FILE mf_personnel
cont> SET TRANSACTION MODES (ALL);
```

2.1.5 Support for Vested Images

OpenVMS VAX OpenVMS Alpha

Oracle Rdb V7.0 no longer supports OpenVMS VAX images translated to OpenVMS Alpha images using the DECmigrate VEST utility. The VEST utility translates OpenVMS VAX images by converting executable and shareable images into functionally equivalent translated images that run on an OpenVMS Alpha system. ♦

2.1.6 RMU Convert Command and System Tables

When the RMU Convert command converts a database from a previous version to Oracle Rdb V7.0, it sets the RDB$CREATED and RDB$LAST_ALTERED columns to the timestamp of the convert operation.

The RDB$xxx_CREATOR columns are set to the current user name (which is space filled) of the converter. Here "xxx" represents the object name, such as in RDB$TRIGGER_CREATOR.

The RMU Convert command also creates the new index on RDB$TRANSFER_RELATIONS if the database is transfer enabled.

2.1.7 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.

The size of the database root file of any given database will increase a minimum of 13 blocks and a maximum of 597 blocks. The actual increase depends mostly on the maximum number of users specified for the database.

2.1.8 Converting from Versions Earlier Than V5.1

You cannot convert databases earlier than V5.1 directly to V7.0. The RMU Convert command for V7.0 supports conversions from V5.1, V6.0, and V6.1 only. If you have a V3.0 through V5.0 database, you must convert it to V5.1, V6.0, or V6.1 and then convert it to V7.0. For example, if you have a V4.2 database, convert it first to V5.1, V6.0, or V6.1. Then, convert the resulting database to V7.0.

If you attempt to convert a database created prior to V5.1 directly to V7.0, Oracle Rdb generates an error.

2.1.9 Functionality Not Available on Digital UNIX

Digital UNIX

This section describes the functionality that is not available yet on Digital UNIX.

The following functionality that affects all interfaces is not available yet on Digital UNIX:

- The Hot Standby option
- Bounded volume sets
- The Ctrl/T keystroke sequence for statistics during a load operation
- Support for operator notification classes

  This support is latent on Digital UNIX. You can specify syntax such as dump or other operator class commands but the operator notification features does not work on Digital UNIX.

  Although this support is latent on Digital UNIX, you can restore a database created on Digital UNIX on an OpenVMS system and operator notification features are immediately activated.

  Support for internal operator notification may be included in a future release.
- Oracle CDD/Repository
- Parallel Backup

  On OpenVMS, you can specify a multiprocess RMU Backup command for backup operations to tape, referred to as a parallel backup. This feature uses multiple, multithreaded processes to perform a database backup. This feature is not available on Digital UNIX.

  Associated with the parallel backup operation, are the RMU Backup Plan command and the Parallel Backup Monitor. Neither the RMU Backup Plan command nor the Parallel Backup Monitor are supported for use with Digital UNIX databases. The RMU Backup command provides qualifiers that allow to you specify a parallel backup operation on OpenVMS; these qualifiers are not valid on Digital UNIX. The qualifiers are: Execute, List_Plan, and Parallel. Likewise, the RMU Backup Plan command and the Parallel Backup Monitor are not available on Digital UNIX.

The following SQL functionality is not available yet on Digital UNIX:

- Some SQL precompiler languages

  The SQL precompiler supports the following languages: Digital UNIX C, DEC C for Digital UNIX, DEC COBOL for Digital UNIX, DEC Fortran, and DEC Pascal.

  Oracle Rdb does not support Ada on Digital UNIX for this release; PL/I is not available on Digital UNIX.
- Some host languages with the SQL module processor

  The SQL module processor supports Digital UNIX C, DEC C for Digital UNIX, DEC COBOL for Digital UNIX, DEC Fortran, and DEC Pascal host languages. Other languages are not supported for this release.
- Creating collating sequences

Oracle Rdb for Digital UNIX does not support the creation of collating sequences with SQL. You can, however, restore a database from OpenVMS and retain the collating sequences that exist in that database. Also, if a Digital UNIX database is altered from an OpenVMS system, collating sequences can be created remotely. All other internationalization features are supported.

- Support for the following logical names:

  SYS$CURRENCY
  SYS$DIGIT_SEP
  SYS$RADIX_POINT
  SYS$LANGUAGE
  RDB$CHARACTER_SET
  RDB$LIBRARY
  RDB$ROUTINES

- Oracle7 SQL functions

  The SQL functions added for compatibility with Oracle7 SQL are not available on Digital UNIX.

- SET LANGUAGE statement

  On Digital UNIX, foreign language support is provided by means of a locale setting. See your system manager to set up locales on Digital UNIX.

- SET DATE FORMAT statement
- SET DICTIONARY statement
- SHOW DICTIONARY, SHOW DATE FORMAT, and SHOW LANGUAGE statements
- Parameter checking

  Parameter checking is not supported on Oracle Rdb for Digital UNIX. (Parameter checking determines whether the SQL module processor compares the number of formal parameters declared for a procedure with the number of parameters specified in the SQL statement. On OpenVMS, you enable parameter checking with the SQL PARAMETER_CHECK qualifier.)

The following Oracle RMU commands are not available yet on Digital UNIX:

- Backup Plan
- Set Audit
- Show Audit

The following Oracle RMU command qualifiers are not supported or are changed for Oracle Rdb for Digital UNIX:

- Qualifiers associated with parallel backup operations:

  The Execute, List_Plan, and Parallel qualifiers of the RMU Backup command are not supported for use on Oracle Rdb for Digital UNIX.

- Qualifiers associated with security auditing:

  – The Audit qualifier of the RMU Load command is not supported on Oracle Rdb for Digital UNIX.

  – The Security option of the Items qualifier of the RMU Extract command is not supported on Oracle Rdb for Digital UNIX.

- The following qualifiers associated with ACL editing in the RMU Set Privilege command:
  - Edit
  - Journal
  - Keep
  - Mode
  - Recover
- Qualifiers associated with operator notification:
  - The Notify qualifier of the RMU Set After_Journal command is not supported on Oracle Rdb for Digital UNIX.
  - The Notify option for the Aij_Options qualifier of the RMU Copy_ Database, Move_Area, Restore, and Restore Only_Root commands is not supported on Oracle Rdb for Digital UNIX.
- Qualifiers associated with data dictionary support:
  - The Path qualifier of the RMU Close, Convert, Open, and Restore commands is not supported on Oracle Rdb for Digital UNIX.
  - The Cdd_Integrate qualifier of the RMU Restore command is not supported on Oracle Rdb for Digital UNIX.
  - The Nocdd_Integrate qualifier of the RMU Restore command is not supported on Oracle Rdb for Digital UNIX.
- Qualifiers associated with VMScluster support:
  - The value of the Nodes_Max qualifier of the RMU Copy_Database, Move_Area, Restore, and Restore Only_Root commands must be 1 on Digital UNIX.
  - The Cluster qualifier of the RMU Close command is not supported on Oracle Rdb for Digital UNIX.
  - The Nocluster qualifier of the RMU Close command is not supported on Oracle Rdb for Digital UNIX.
- A qualifier associated with tape support:

  The Density qualifier of the RMU Backup, Backup After_Journal, and Optimize After_Journal commands is not supported on Oracle Rdb for Digital UNIX.
- Qualifiers associated with OpenVMS processes:
  - The Priority qualifier of the RMU Monitor Start command is not supported on Oracle Rdb for Digital UNIX.
  - The Swap qualifier of the RMU Monitor Start command is not supported on Oracle Rdb for Digital UNIX.
- The Abort=FORCEX and Abort=DELPRC qualifiers for the RMU Close and Monitor Stop commands

  These qualifiers of the Close and Monitor Stop commands on Oracle Rdb for OpenVMS are replaced with the Abort qualifier on Oracle Rdb for Digital UNIX. The Abort qualifier on Oracle Rdb for Digital UNIX exhibits behavior equivalent to the Abort=DELPRC qualifier on Oracle Rdb for

OpenVMS. (You can use the Abort=FORCEX and Abort=DELPRC qualifiers on Oracle Rdb for Digital UNIX, but both exhibit the behavior associated with the Abort=DELPRC qualifier.)

- The Language=RDO qualifier for the RMU Extract command

  There is no RDO support on Oracle Rdb for Digital UNIX.

- The Disk qualifier for the RMU Set Corrupt_Pages command

  The Disk qualifier of the RMU Set Corrupt_Pages command is replaced with the File_System=path qualifier on Oracle Rdb for Digital UNIX. When you specify the File_System qualifier with the RMU Set Corrupt_Pages command on Oracle Rdb for Digital UNIX, any areas with the same mount point as the specified directory path are set as you indicate with the Corrupt or Consistent qualifiers.

- Symbol settings

  The setting of symbols by the RMU Set After_Journal, Show After_Journal, and Show Version commands on Oracle Rdb for OpenVMS are not supported by these commands on Oracle Rdb for Digital UNIX.

  On Oracle Rdb for OpenVMS, the RMU Backup After_Journal, Set After_Journal, and Show After_Journal commands set process global symbols. No equivalent behavior is supported on Oracle Rdb for Digital UNIX.

  In addition, the RMU Show Version command on OpenVMS sets the following symbols: RMU$RDB_VERSION and RMU$DATABASE_VERSION. No equivalent behavior is supported on Oracle Rdb for Digital UNIX.

Note

The names of a few Oracle RMU qualifiers have been changed in Oracle Rdb for Digital UNIX to remove references specific to OpenVMS. For example, the Rms_Record_Definition qualifier for Oracle Rdb for OpenVMS has changed to the Record_Definition qualifier for Oracle Rdb for Digital UNIX. However, an application migrated from OpenVMS to Digital UNIX that contains one of these qualifiers need not be changed. Oracle RMU on Digital UNIX recognizes the OpenVMS qualifiers unless otherwise noted in this section. (Likewise, Oracle RMU for OpenVMS recognizes the Oracle Rdb for Digital UNIX qualifiers.)

♦

2.1.10 Record Caches and Exclusive Access

If a table has a row-level cache defined for it, the Record 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.

2.1.11 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. For example:

```
SQL> SELECT * FROM T1 WHERE ID >= 401;
Conjunct        Get     Retrieval sequentially of relation T1
Strict Partitioning: part       3
          ID   LAST_NAME      FIRST_NAME
         450   Gentile        Russ
1 row selected
```

2.1.12 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 existing page size and the database has been manually opened or users are active, the add operation fails with the following error:

```
%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. The reason for this restriction is that several internal Oracle Rdb data structures are based on the minimum page size and these structures cannot be resized if there are users attached to the database. Furthermore, because this particular change is not recorded in the AIJ, any recovery scenario will fail.

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.

2.2 SQL Known Problems and Restrictions

This section describes known problems and restrictions for the SQL interface for Version 7.0.

2.2.1 Behavior of Journaling Using IMPORT

The IMPORT statement does not let you restore original journal settings. Because IMPORT disables journaling, FAST COMMIT is disabled. Thus, the resulting database is not in its original state.

Oracle Rdb does not let you enable after-image journaling with the CREATE DATABASE statement. Because the IMPORT statement shares a common code path with the CREATE DATABASE statement, the after-image journal attributes cannot be imported. Therefore, the after-image journal attributes are disabled after IMPORT, inherently implying that FAST COMMIT is also disabled.

This restriction will be investigated for a future release.

You can use the following workaround:

Before exporting the database, use the following RMU Extract command to generate a script of the after-image journal definition. After the database is exported and imported, run the script to re-create the original after-image journal settings:

```
$ RMU/EXTRACT/ITEM=ALTER_DATABASE/OUTPUT=ADD_AIJ.SQL yourdatabase
```

2.2.2 Cannot Alter a Storage Map That Is Vertically Partitioned

You cannot alter a storage map that is vertically partitioned; you cannot modify a storage map that is not partitioned vertically to one that is partitioned vertically.

The following example shows the error message that Oracle Rdb returns:

```
SQL> ALTER STORAGE MAP T_MAP ENABLE COMPRESSION;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-WISH_LIST, feature not implemented yet
-RDMS-E-VRPINVALID, invalid operation for storage map "T_MAP"
```

This restriction may be lifted in a future version of Oracle Rdb. Currently, the recommended method of altering the storage map is to unload and load the table.

2.2.3 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 storeage 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> --
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

SQL>
```

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.

2.2.4 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.)

2.2.5 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 or configuration parameter. This note describes the differences between the two mechanism.

- If you define the RDMS$BIND_QG_REC_LIMIT logical name or RDB_BIND_QG_REC_LIMIT configuration parameter to a small value, the query will often fail 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 tables 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 table) is sufficient to read each column definition.

  To see an indication of the queries executed against the system tables, define the RDMS$DEBUG_FLAGS logical name or the RDB_DEBUG_FLAGS configuration parameter 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 or the configuration parameter RDB_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 tables as part of query processing.

2.2.6 Suggestions for Optimal Usage 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. (See Section 2.2.7 for more information about index names.)
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 table 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$BIND_BUFFERS logical name or RDB_BIND_BUFFERS configuration parameter or the NUMBER OF BUFFERS IS clause in SQL to change the number of buffers).
- To distribute the disk I/O load, store the storage areas for the indexes on separate disk drives. Note that using the same storage area for multiple indexes will result in contention during the index creation (Step 5) for SPAM pages.
- Consider placing the .ruj file for each parallel definer on its own disk or an infrequently used disk.
- Even though snapshot I/O should be minimal, consider disabling snapshots during parallel index creation.
- Refer to the *Oracle Rdb7 Guide to Database Performance and Tuning* to determine the appropriate working set values for each process to minimize excessive paging activity. In particular, avoid using working set parameters where the difference between WSQUOTA and WSEXTENT is large. The SORT utility uses the difference between these two values to allocate scratch virtual memory. A large difference (that is, the requested virtual memory grossly exceeds the available physical memory) may lead to excessive page faulting.
- The performance benefits of using SHARED DATA DEFINITION can best be observed when creating many indexes in parallel. The benefit is in the average elapsed time, not in CPU or I/O usage. For example, when two indexes are created in parallel using the SHARED DATA DEFINITION clause, the database must be attached twice, and the two attaches each use separate system resources.
- Using the SHARED DATA DEFINITION clause on a single-file database or for indexes defined in the RDB$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.

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

2.2.7 %SQL-F-IND_EXISTS During Concurrent Index Definition

During concurrent index creation, Oracle Rdb validates only the first 27 bytes of the index name. Therefore, these bytes must be unique. After the index creation is complete, Oracle Rdb uses the full index name. This restriction happens only when multiple index definitions are in progress.

To permit concurrent index definitions, Oracle Rdb takes out a special lock on the provided index name. The lock manager allows 31 bytes for lock names, however Oracle Rdb needs 4 bytes for database context, thus leaving only 27 bytes to keep the index names unique. Therefore, index names must be unique to 27 bytes when you create indexes concurrently.

The following example demonstrates this problem:

```
Session 1:
SQL> CREATE INDEX EMP_DAILY_SALES_SUMRY_SORTED_2
cont>       ON EMPLOYEES (ADDRESS_DATA_2);

Session 2:
SQL> CREATE INDEX EMP_DAILY_SALES_SUMRY_SORTED_1
cont>       ON EMPLOYEES (ADDRESS_DATA_1);
%SQL-F-IND_EXISTS, Index EMP_DAILY_SALES_SUMRY_SORTED_1 already exists in this
database or schema
```

2.2.8 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>     lang 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.

2.2.9 Incorrect Processing of Subquery When Nested in FOR Cursor Loop

A subquery may return incorrect results when it appears in a SET statement nested within a FOR cursor loop and this subquery refers to local variables initialized inside the FOR cursor loop.

This problem is due to an optimization which pulls the subquery evaluation into the FOR cursor loop's own query, thereby evaluating it before the local variables (or parameters) have been initialized.

The following example shows the problem:

```
SQL> set flags 'TRACE';
SQL>
SQL> begin
cont> declare :id char(5);
cont> declare :sal integer(2);
cont>
cont> for :emp as
cont>     select last_name, employee_id
cont>     from employees
cont>     where employee_id = '00164'
cont> do
cont>     set :id = :emp.employee_id;
cont>     set :sal = (select salary_amount
cont>           from salary_history
cont>           where employee_id = :id
cont>               and salary_end is null);
cont>     trace 'Employee: ', :id, ', Salary: ', :sal;
cont> end for;
cont> end;
~Xt: Employee: 00164, Salary: 0.00
```

The salary should not be zero. This incorrect value is returned because the subquery requires the local variable ID, which is assigned a value within the FOR loop prior to the subquery. However, this assignment to ID is performed after the subquery has been evaluated.

A workaround is to reference the FOR loop columns directly using the cursors handle, rather than taking copies before the subquery is executed.

```
SQL> begin
cont> declare :id char(5);
cont> declare :sal integer(2);
cont>
cont> for :emp as
cont>     select last_name, employee_id
cont>     from employees
cont>     where employee_id = '00164'
cont> do
cont>     set :id = :emp.employee_id;
cont>     set :sal = (select salary_amount
cont>           from salary_history
cont>           where employee_id = :emp.employee_id
cont>               and salary_end is null);
cont>     trace 'Employee: ', :id, ', Salary: ', :sal;
cont> end for;
cont> end;
~Xt: Employee: 00164, Salary: 51712.00
```

The correct result is returned when you use the FOR loop handle and a direct column reference.

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

2.2.10 Nested Correlated Subquery Outer References Incorrect

Outer references from aggregation subqueries contained within nested queries could receive incorrect values, causing the overall query to return incorrect results. The general symptom for an outer query that returned rows 1 to n was that the inner aggregation query would operate with the nth - 1 row data (usually NULL for row 1) when it should have been using the nth row data.

This problem has existed in various forms for all previous versions of Oracle Rdb, but only appears in V6.1 and later when the inner of the nested queries contains an UPDATE statement.

The following example demonstrates the problem:

```
SQL> attach 'filename shipping';
SQL> select * from manifest where voyage_num = 4904 or
cont>                             voyage_num = 4909;
  VOYAGE_NUM       EXP_NUM   MATERIAL              TONNAGE
        4904           311   CEDAR                    1200
        4904           311   FIR                       690
        4909           291   IRON ORE                 3000
        4909           350   BAUXITE                  1100
        4909           350   COPPER                   1200
        4909           355   MANGANESE                 550
        4909           355   TIN                       500
7 rows selected
SQL> begin
cont> for :a as each row of
cont>  select * from voyage v where v.ship_name = 'SANDRA C.' or
cont>                               v.ship_name = 'DAFFODIL' do
cont>   for :b as each row of table cursor modcur1 for
cont>    select * from  manifest m where m.voyage_num = :a.voyage_num do
cont>     update manifest
cont>      set tonnage = (select (avg (m1.exp_num) *3) from manifest m1
cont>                     where m1.voyage_num = :a.voyage_num)
cont>      where current of modcur1;
cont>   end for;
cont> end for;
cont> end;
SQL> select * from manifest where voyage_num = 4904 or
cont>                              voyage_num = 4909;
  VOYAGE_NUM       EXP_NUM   MATERIAL              TONNAGE
        4904           311   CEDAR                    NULL
        4904           311   FIR                      NULL
        4909           291   IRON ORE                  933
        4909           350   BAUXITE                   933
        4909           350   COPPER                    933
        4909           355   MANGANESE                 933
        4909           355   TIN                       933
7 rows selected
```

The correct value for TONNAGE on both rows for VOYAGE_NUM 4904 (outer query row 1) is: AVG (311 + 311) *3 = 933. However, Oracle Rdb calculates it as: AVG (NULL + NULL) *3 = NULL. In addition, the TONNAGE value for VOYAGE_NUM 4909 (outer query row 2) is actually the TONNAGE value for outer query row 1.

A workaround is to declare a variable of the same type as the outer reference data item, assign the outer reference data into the variable before the inner query that contains the correlated aggregation subquery, and reference the variable in the aggregation subquery. Keep in mind the restriction on the use of local variables in FOR cursor loops described by Section 2.2.9.

For example:

```
SQL> declare :vn integer;
SQL> begin
cont> for :a as each row of
cont>  select * from voyage v where v.ship_name = 'SANDRA C.' do
cont>   set :vn = :a.voyage_num;
cont>   for :b as each row of table cursor modcur1 for
cont>    select * from manifest m where m.voyage_num = :a.voyage_num do
cont>     update manifest
cont>      set tonnage = (select (avg (m1.exp_num) *3) from manifest m1
cont>                    where m1.voyage_num = :vn)
cont>      where current of modcur1;
cont>   end for;
cont> end for;
cont> end;
SQL> select * from manifest where voyage_num = 4904;
  VOYAGE_NUM      EXP_NUM   MATERIAL              TONNAGE
        4904          311   CEDAR                     933
        4904          311   FIR                       933
```

This problem will be corrected in a future release of Oracle Rdb.

2.2.11 Additional Usage Notes for 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 (assume that the data is sorted) 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.

  For example, 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$BIND_HOLD_CURSOR_SNAP or configuration parameter RDB_BIND_HOLD_CURSOR_SNAP to the value 1 to force all hold cursors to fetch the result set into a cached data area. (The cached data area appears as a “Temporary Relation” in the optimizer strategy displayed by the SET FLAGS 'STRATEGY' statement or the RDMS$DEBUG_FLAGS "S" flag.) This logical name or configuration parameter helps to stabilize the cursor to some degree.

2.3 Oracle RMU Known Problems and Restrictions

This section describes known problems and restrictions for the RMU interface for Version 7.0.

2.3.1 Default for RMU Checksum and CRC Qualifiers Changing in Future Release

The default behavior for the Checksum_Verification and Crc qualifiers for the following RMU commands will be changed in a future release of Oracle Rdb:

- Backup
- Backup After_Journal
- Backup Plan
- Optimize After_Journal

Currently, the default value for the CRC qualifier is Crc=Autodin_II for NRZ /PE (800/1600 bits/inch) tape drives; Crc=Checksum is the default for GCR (6250 bits/inch) tape drives and for TA78, TA79, and TA81 tape drives; and Nocrc is the default for TA90 (IBM 3480 class) drives. In a future release, the default value for the CRC qualifier will be Crc=Checksum for all tape drives except NRZ/PE (800/1600 bits/inch) tape drives. The default qualifier for the NRZ/PE (800/1600 bits/inch) tape drives will remain Crc=Autodin_II. The Crc=Checksum qualifier verifies the checksum on each buffer of data before it is written to tape or disk. This provides end-to-end error detection for the backup file I/O.

Currently, for the Backup commands shown in the previous list, the default value for the Checksum_Verification qualifier is Nochecksum_Verification. In a future release, the default qualifier will be Checksum_Verification. The Checksum_Verification qualifier requests that the Oracle RMU command verify the checksum stored on each database page before the Oracle RMU backup operation is applied, thereby providing end-to-end error detection on the database I/O.

Oracle Corporation recommends that you accept the new default behaviors for your applications. They prevent you from including corrupt database pages in backup files and optimized .aij files. Without the checksum verifications, corrupt data pages in these files will not be detected when the files are restored. The corruptions on the restored page may not be detected until weeks or months after the backup file is created, or it is possible the corruption may not be detected at all.

However, if you require the behavior of the Nocrc or Nochecksum_Verification qualifiers and are willing to risk the undetected corruptions, this advance notice is given so that you can prepare for the change in the default behavior.

2.3.2 Performance Monitor Collection Cells Are Reused

Previous obsolete collection cells used in the Performance Monitor are reused in V7.0. As a result, repository definitions of the global sections from versions prior to V7.0 are usable, but not necessarily correct for V7.0.

OpenVMS VAX OpenVMS Alpha

On OpenVMS, to avoid problems, especially if you use an alternative application to display statistics, execute the SYS$LIBRARY:RMU$SHOW_ STATISTICSnn.CDO command procedure after you convert to V7.0. ♦

2.3.3 Collect Optimizer Statistics After Converting a Database to V7.0

Oracle Corporation recommends that you execute an RMU Collect Optimizer_ Statistics Command after you convert an Oracle Rdb database to V7.0.

Prior to this release, Oracle Rdb maintained cardinalities for tables and indexes; multisegment sorted index prefix cardinalities were not collected. In addition, the stored cardinality values could differ from the actual cardinality values if the RDB$SYSTEM storage area had been set to read-only access.

When you convert an Oracle Rdb database to V7.0, cardinalities currently stored for tables and indexes are retained for the converted database. In addition, the RMU Convert command estimates the prefix cardinalities for each multisegment sorted index.

By issuing the RMU Collect Optimizer_Statistics command after converting a database to Oracle Rdb V7.0, you ensure that the table and index cardinality values your converted database uses are accurate. Inaccurate cardinality values can result in poor query performance.

The following example shows the command to use after converting a database if the database being converted has the RDB$SYSTEM storage area set to read-only:

```
$ RMU/COLLECT OPTIMIZER_STATISTICS/CARDINALITY my_db.rdb
```

The following example shows the command to use after converting a database if the RDB$SYSTEM storage area is *not* set to read-only. In this example, only the prefix cardinalities are updated from estimated to actual values. The cardinalities of table and non-multisegement sorted indexes are not updated:

```
$ RMU/COLLECT OPTIMIZER_STATISTICS/CARDINALITY -
_$ /INDEX=(multisegement-index-name-list)/NOTABLE my_db.rdb
```

The syntax shown in the second example results in a quicker RMU Collect Optimizer_Statistics operation.

See the *Oracle RMU Reference Manual* for a complete description of the RMU Collect Optimizer_Statistics command.

2.3.4 RMU Parallel Backup Command Not Supported for Use with SLS

OpenVMS VAX OpenVMS Alpha

The RMU Parallel Backup command is not supported for use with the Storage Library System (SLS) for OpenVMS. ♦

2.3.5 RMUwin, Rdb Performance Monitor Limit Motif Support

RMUwin and the Oracle Rdb Performance Monitor for Oracle Rdb V7.0 do not support the DECwindows Motif software interface. Oracle Rdb V7.0 supports these tools only on windows software interfaces. (See Section 1.5.3 for more information about the V7.0 windows support.) However, you can continue to use the Motif interface to RMUwin and the Oracle Rdb Performance Monitor to access Oracle Rdb V6.1 databases.

For example, if you have the multiversion variant of V6.1 installed, you can continue to use the Motif interfaces of RMUwin and the Performance Monitor against V6.1 databases. However, if you install standard Oracle Rdb V7.0, you will no longer be able to use these Motif interfaces.

2.4 Known Problems and Restrictions in All Interfaces for Version 6.1 and Earlier

The following problems and restrictions from Version 6.1 and earlier still exist.

2.4.1 Restriction on Tape Usage for Digital UNIX V3.2

Digital UNIX

You can experience a problem where you are unable to use multiple tapes with the Oracle RMU Backup command with Digital UNIX V3.2. Every attempt to recover fails. If this happens and device errors are logged in the system error log, you may have encountered the following situation:

If an error is detected by Digital UNIX during the open operation of the tape device, it is possible that the operation succeeded but the device open reference count is zeroed out. This means that any attempt to use the drive by the process holding the open file descriptor will fail with EINVAL status but another process will be able to open and use the drive even while the first process has it opened.

There is no workaround for this problem. This problem with the magtape driver will be corrected in a future release of Digital UNIX. ♦

2.4.2 Support for Single-File Databases to Be Dropped in a Future Release

Oracle Rdb currently supports both single-file and multifile databases on both OpenVMS and Digital UNIX. 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 in Oracle Rdb V6.1 and V7.0.

2.4.3 DECdtm Log Stalls

OpenVMS VAX OpenVMS Alpha

Resource managers using the DECdtm services sometimes suddenly stop being able to commit transactions. The systems have been running fine for some period of time, but suddenly they stop. If Oracle Rdb is installed and trying to run transactions, an RMU Show command on the affected database will show transactions as being "stalled, waiting to commit".

Refer to the DECdtm documentation and release notes for information on symptoms, fixes, and workarounds to this problem. One workaround, for OpenVMS V5.5-x, is provided here.

On the affected node, and while the log stall is in progress, perform the following command from a privileged account:

```
$ MCR LMCP SET NOTIMEZONE
```

This should force the log to restart.

This stall occurs only when a particular bit in a pointer field becomes set. To see the value of the pointer field, enter the following command from a privileged account (where <nodename> is the SCS node name of the node in question).

```
$ MCR LMCP DUMP/ACTIVE/NOFORM SYSTEM$<nodename>
```

This command displays output similar to the following:

```
Dump of transaction log SYS$COMMON:[SYSEXE]SYSTEM$<nodename>.LM$JOURNAL;1
End of file block 4002 / Allocated 4002
Log Version 1.0
Transaction log UID:   29551FC0-CBB7-11CC-8001-AA000400B7A5
Penultimate Checkpoint: 000013FD4479 0079
Last Checkpoint:        000013FDFC84 0084

Total of 2 transactions active, 0 prepared and 2 committed.
```

The stall will occur when bit 31 of the checkpoint address becomes set, as this excerpt from the previous example shows:

```
        Last Checkpoint:        000013FDFC84 0084
                                    ^
                                    |
```

When the number indicated in the example becomes 8, the log will stall. Check this number and observe how quickly it grows. When it is at 7FFF, frequently use the following command:

```
$ MCR LMCP SHOW LOG /CURRENT
```

If this command shows a stall in progress, use the workaround to restart the log.

See your Digital representative for information about patches to DECdtm. ♦

2.4.4 You Cannot Run Distributed Transactions on Systems with DECnet/OSI and OpenVMS Alpha Version 6.1 or OpenVMS VAX Version 6.0

OpenVMS VAX OpenVMS Alpha

If you have DECnet/OSI installed on a system with OpenVMS Alpha Version 6.1 or OpenVMS VAX Version 6.0, you cannot run Oracle Rdb operations that require the two-phase commit protocol. The two-phase commit protocol guarantees that if one operation in a distributed transaction cannot be completed, none of the operations is completed.

If you have DECnet/OSI installed on a system running OpenVMS VAX Version 6.1 or higher or OpenVMS Alpha V6.2 or higher, you can run Oracle Rdb operations that require the two-phase commit protocol.

For more information about the two-phase commit protocol, see the *Oracle Rdb7 Guide to Distributed Transactions*. ♦

2.4.5 Multiblock Page Writes May Require Restore Operation

OpenVMS VAX OpenVMS Alpha

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 will not be 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. ♦

2.4.6 Oracle Rdb Network Link Failure Does Not Allow DISCONNECT to Clean Up Transactions

If a program attaches to a database on a remote node and it loses the connection before the COMMIT statement is issued, there is nothing you can do except exit the program and start again.

The problem occurs when a program is connected to a remote database and updates the database, but then just before it commits, the network fails. When the commit executes, SQL shows, as it normally should, that the program has lost the link. Assume that the user waits for a minute or two, then tries the transaction again. The problem is that when the start transaction is issued for the second time, it fails because old information still exists about the previous failed transaction. This occurs even if the user issues a DISCONNECT statement (in V4.1 and earlier, a FINISH statement), which also fails with an RDB-E-IO_ERROR error message.

2.4.7 Replication Option Copy Processes Do Not Process Database Pages Ahead of an Application

OpenVMS VAX OpenVMS Alpha

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 will catch up to the application and will not be able to process database pages that are logically ahead of the application in the RDB$CHANGES system table. 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$TRANSFERS system table and then tries to delete any RDB$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.

This condition is being investigated further in a future release of Oracle Rdb. ♦

2.5 SQL Known Problems and Restrictions for Oracle Rdb Version 6.1 and Earlier

The following problems and restrictions from Oracle Rdb Version 6.1 and earlier still exist.

2.5.1 INCLUDE SQLDA2 Statement Is Not Supported for SQL Precompiler for PL/I in Oracle Rdb V5.0 or Higher

OpenVMS VAX OpenVMS Alpha The SQL statement INCLUDE SQLDA2 is not supported for use with the PL/I precompiler in Oracle Rdb V5.0 or higher.

There is no workaround. This problem will be fixed in a future version of Oracle Rdb. ♦

2.5.2 SQL Pascal Precompiler Processes ARRAY OF RECORD Declarations Incorrectly

OpenVMS VAX OpenVMS Alpha The Pascal precompiler for SQL gives an incorrect %SQL-I-UNMATEND error when it parses a declaration of an array of records. The precompiler does not associate the END statement with the record definition, and the resulting confusion in host variable scoping causes a fatal error.

To avoid the problem, declare the record as a type and then define your array of that type. For example:

```
    main.spa:

        program main (input,output);

        type
        exec sql include 'bad_def.pin';    !gives error
        exec sql include 'good_def.pin';   !ok
        var
            a : char;

        begin
        end.

---------------------------------------------------------------
    bad_def.pin

    x_record = record
    y  : char;
    variable_a:  array [1..50] of record
                a_fld1 : char;
                b_fld2  : record;
                            t : record
                                      v : integer;
                            end;
                end;
        end;
    end;
 ---------------------------------------------------------------

    good_def.pin

good_rec = record
        a_fld1 : char;
        b_fld2 : record
                t : record
                        v: integer;
                end;
        end;
end;
```

```
x_record = record
    y  : char
    variable_a : array [1..50] of good_rec;
end;                                                       ♦
```

2.6 Oracle RMU Known Problems and Restrictions for Oracle Rdb Version 6.1 and Earlier

The following and restrictions problems from Oracle Rdb Version 6.1 and earlier still exist.

2.6.1 Oracle RMU Commands Pause During Tape Rewind

Digital UNIX

For Oracle Rdb V6.1 or higher on Digital UNIX, the Oracle RMU Backup and Restore commands pause under certain conditions.

If multiple tape drives are used for RMU Backup or RMU Restore commands and a tape needs to rewind, the Oracle RMU command pauses until the rewind is complete. This is different from behavior on OpenVMS systems where the command continues to write to tape drives that are not rewinding.

There is no workaround for this problem. ♦

2.6.2 TA90 and TA92 Tape Drives Are Not Supported on Digital UNIX

Digital UNIX

When rewinding or unloading tapes using either TA90 and TA92 drives, Digital UNIX intermittently returns an EIO error, causing the Oracle RMU operation to abort. This problem occurs most often when Oracle RMU accesses multiple tape drives in parallel. However, the problem occurs even with single-tape drive access.

As a result of this problem, Oracle Rdb for Digital UNIX supports neither TA90 nor TA92 tape drives. ♦

2.7 RDML Known Problems and Restrictions for Version 7.0 and Earlier

OpenVMS VAX OpenVMS Alpha

The following problems exist in Version 7.0.

2.7.1 RDML Generates Undefined Symbol at Link Time Using Multiversion Oracle Rdb

When the multiversion variant of Oracle Rdb V5.1 and later versions are installed, Pascal code containing RDML statements may generate undefined symbols at link time. This occurs when the code inherits a Pascal environment file that also contains RDML statements and the code is compiled with the /DEBUG qualifier against a V4.0 database. The following errors are generated:

```
%LINK-W-NUDFSYMS, 2 undefined symbols:
%LINK-I-UDFSYM,         RDML$VPAS_INITIALIZE2
%LINK-I-UDFSYM,         RDML$VPAS_START_TRANS2
```

This problem is caused by using a non-varianted version of the RDMLVPAS.PAS file. RDML may add new versions of files for new releases. For example, RDML added new functions RDML$VPAS_INITIALIZE2 and RDML$VPAS_START_TRANS2 for V5.1 and RDML$VPAS_INITIALIZE3 for V6.0. If you use multiversion, always refer to the latest version of the RDMLVPAS.PAS.

To avoid this problem, always link against the latest version of RDMLRTL.OLB for multiversion support. Alternatively, you can use the DCL command LIBR /EXTRACT=(RDML_VPAS_SUPPORT) to extract the object file for RDML$VPAS_INITIALIZE2, RDML$VPAS_INITIALIZE3 and RDML$VPAS_START_TRANS2 and then use the LIBR/REPLACE command to manually insert it into the RDMLRTL.OLB for V4.2 and V5.1, as shown in the following example:

```
$ library/extract=RDML_VPAS_SUPPORT/out=vpas.obj sys$library:rdmrtl60
$ library/replace/log sys$library:rdmrtl42 vpas.obj
```

♦

2.8 Oracle CDD/Repository Notes of General Interest

OpenVMS VAX OpenVMS Alpha This section describes notes of general interest, including known problems and restrictions for the repository for Version 7.0, including problems first seen in previous versions.

2.8.1 Oracle CDD/Repository Compatibility with Oracle Rdb Features

Some Oracle Rdb features are not fully supported by all versions of Oracle CDD/Repository. Table 2–1 shows which versions of Oracle CDD/Repository support Oracle Rdb features and the extent of support.

In Table 2–1, repository support for Oracle Rdb features can vary as follows:

- Explicit support—The repository recognizes and integrates the feature, and you can use the repository to manipulate the item.
- Implicit support—The repository recognizes and integrates the feature, but you cannot use any repository interface to manipulate the item.
- Pass-through support—The repository does not recognize or integrate the feature, but allows the Oracle Rdb operation to complete without aborting or overwriting metadata. With pass-through support, a CDD-I-MBLRSYNINFO informational message may be returned.

Table 2–1 Oracle CDD/Repository Compatibility for Oracle Rdb Features

Oracle Rdb Feature	Minimum Version of Oracle Rdb	Minimum Version of Oracle CDD/Repository	Support
CASE, NULLIF, and COALESCE expressions	V6.0	V6.1	Implicit
CAST function	V4.1	V7.0	Explicit
Character data types to support character sets	V4.2	V6.1	Implicit
Collating sequences	V3.1	V6.1	Explicit
Constraints (PRIMARY KEY, UNIQUE, NOT NULL, CHECK, FOREIGN KEY)	V3.1	V5.2	Explicit
CURRENT_DATE, CURRENT_TIME, and CURRENT_TIMESTAMP functions	V4.1	V7.0	Explicit

(continued on next page)

Table 2–1 (Cont.) Oracle CDD/Repository Compatibility for Oracle Rdb Features

Oracle Rdb Feature	Minimum Version of Oracle Rdb	Minimum Version of Oracle CDD/Repository	Support
CURRENT_USER, SESSION_USER, SYSTEM_USER functions	V6.0	V7.0	Explicit
Date arithmetic	V4.1	V6.1	Pass-through
DATE ANSI, TIME, TIMESTAMP, and INTERVAL data types	V4.1	V6.1	Explicit
Delimited identifiers	V4.2	V6.1[1]	Explicit
External functions	V6.0	V6.1	Pass-through
External procedures	V7.0	V6.1	Pass-through
EXTRACT, CHAR_LENGTH, and OCTET_LENGTH functions	V4.1	V6.1	Explicit
GRANT/REVOKE privileges	V4.0	V5.0 accepts but does not store information	Pass-through
Indexes	V1.0	V5.2	Explicit
INTEGRATE DOMAIN	V6.1	V6.1	Explicit
INTEGRATE TABLE	V6.1	V6.1	Explicit
Logical area thresholds for storage maps and indexes	V4.1	V5.2	Pass-through
Multinational character set	V3.1	V4.0	Explicit
Multiversion environment (multiple Rdb versions)	V4.1	V5.1	Explicit
NULL keyword	V2.2	V7.0	Explicit
Oracle7 compatibility functions, such as CONCAT, CONVERT, DECODE, and SYSDATE	V7.0	V7.0	Explicit
Outer joins, derived tables	V6.0	V7.0	Pass-through
Query outlines	V6.0	V6.1	Pass-through
Storage map definitions correctly restored	V3.0	V5.1	Explicit
Stored functions	V7.0	V6.1	Pass-through
Stored procedures	V6.0	V6.1	Pass-through
SUBSTRING function	V4.0	V7.0 supports all features V5.0 supports all but V4.2 MIA features [2]	Explicit
Temporary tables	V7.0	V6.1	Pass-through
Triggers	V3.1	V5.2	Pass-through
TRUNCATE TABLE	V7.0	V6.1	Pass-through

[1]The repository does not preserve the distinction between uppercase and lowercase identifiers. If you use delimited identifiers with Oracle Rdb, the repository ensures that the record definition does not include objects with names that are duplicates except for case.

[2]Multivendor Integration Architecture (MIA) features include the CHAR_LENGTH clause and the TRANSLATE function.

(continued on next page)

Table 2–1 (Cont.) Oracle CDD/Repository Compatibility for Oracle Rdb Features

Oracle Rdb Feature	Minimum Version of Oracle Rdb	Minimum Version of Oracle CDD/Repository	Support
TRIM and POSITION functions	V6.1	V7.0	Explicit
UPPER, LOWER, TRANSLATE functions	V4.2	V7.0	Explicit
USER function	V2.2	V7.0	Explict

2.9 Oracle CDD/Repository Restrictions for Oracle RdbV7.0 and Earlier

This section describes known problems and restrictions in Oracle CDD/Repository V7.0 and earlier.

2.9.1 Multischema Databases and CDD/Repository

You cannot use multischema databases with CDD/Repository and Oracle Rdb V7.0 and earlier. This problem will be corrected in a future release of Oracle Rdb.

2.9.2 Interaction of Oracle CDD/Repository V5.1 and Oracle RMU Privileges Access Control Lists

OpenVMS VAX

Oracle Rdb provides special Oracle RMU privileges that use the unused portion of the OpenVMS access control list (ACL) to manage access to Oracle RMU operations.

You can use the RMU Set Privilege and RMU Show Privilege commands to set and show the Oracle RMU privileges. The DCL SHOW ACL and DIRECTORY /ACL commands also show the added access control information; however, these tools cannot translate the names defined by Oracle Rdb.

Note

The RMU Convert command propagates the database internal ACL to the root file for access control entries (ACEs) that possess the SECURITY and DBADM (ADMINISTRATOR) privileges.

Oracle CDD/Repository protects its repository (dictionary) by placing the CDD$SYSTEM rights identifier on each file created within the anchor directory. CDD$SYSTEM is a specially reserved rights identifier created by Oracle CDD/Repository.

When Oracle CDD/Repository executes the DEFINE REPOSITORY command, it adds (or augments) an OpenVMS default ACL to the anchor directory. Typically, this ACL allows access to the repository files for CDD$SYSTEM and denies access to everyone else. All files created in the anchor directory inherit this default ACL, including the repository database.

Unfortunately, there is an interaction between the default ACL placed on the repository database by Oracle CDD/Repository and the Oracle RMU privileges ACL processing.

Within the ACL on the repository database, the default access control entries (ACEs) that were inherited from the anchor directory will precede the ACEs added by RMU Restore. As a result, the CDD$SYSTEM identifier will not have any Oracle RMU privileges granted to it. Without these privileges, if the user does not have the OpenVMS SYSPRV privilege enabled, Oracle RMU operations, such as Convert and Restore, will not be allowed on the repository database.

The following problems may be observed by users who do not have the SYSPRV privilege enabled:

- While executing a CDO DEFINE REPOSITORY or DEFINE DICTIONARY command:
  - If the CDD$TEMPLATEDB backup (.rbf) file was created by a previous version of Oracle Rdb, the automatic RMU Convert operation that will be carried out on the .rbf file will fail because SYSPRV privilege is required.
  - If the CDD$TEMPLATEDB backup (.rbf) file was created by the current version of Oracle Rdb, the restore of the repository database will fail because the default ACEs that already existed on the repository file that was backed up will take precedence, preventing RMU$CONVERT and RMU$RESTORE privileges from being granted to CDD$SYSTEM or the user.
  - If no CDD$TEMPLATEDB is available, the repository database will be created without a template, inheriting the default ACL from the parent directory. The ACE containing all the required Oracle RMU privileges will be added to the end of the ACL; however, the preexisting default ACEs will prevent any Oracle RMU privilege from being granted.
- You must use the RMU Convert command to upgrade the database disk format to Oracle Rdb V7.0 after installing V7.0. This operation requires the SYSPRV privilege.

  During the conversion, RMU Convert adds the ACE containing the Oracle RMU privileges at the end of the ACL. Because the repository database already has the default Oracle CDD/Repository ACL associated with it, the Oracle CDD/Repository ACL will take precedence, preventing the granting of the Oracle RMU privileges.
- During a CDO MOVE REPOSITORY command, the Oracle RMU privilege checking may prevent the move, as the RMU$COPY privilege has not been granted on the repository database.
- When you execute the CDD template builder CDD_BUILD_TEMPLATE, the step involving RMU Backup privilege has not been granted.

Oracle CDD/Repository Versions 5.2 and higher correct this problem. A version of the Oracle CDD/Repository software that corrects this problem and allows new repositories to be created using Oracle Rdb V7.0 is provided on the Oracle Rdb V7.0 kit for use on OpenVMS VAX systems. See Section 2.9.2.1 for details.

2.9.2.1 Installing the Corrected CDDSHR Images

OpenVMS VAX

Note

The following procedure must be carried out if you have installed or plan to install Oracle Rdb for OpenVMS VAX V7.0 and have already installed CDD/Repository V5.1 software on your system.

Due to the enhanced security checking associated with Oracle RMU commands in Oracle Rdb for OpenVMS VAX, existing CDDSHR images for CDD/Repository V5.1 must be upgraded to ensure that the correct Oracle RMU privileges are applied to newly created or copied repository databases.

Included in the Oracle Rdb Version 7.0 for OpenVMS VAX distribution kit is a CDD upgraded image kit, called CDDRDB042, that must be installed after you have installed the Oracle Rdb Version 7.0 for OpenVMS VAX kit.

This upgrade kit should be installed by using VMSINSTAL. It automatically checks which version of CDDSHR you have installed and replaces the existing CDDSHR.EXE with the corrected image file. The existing CDDSHR.EXE will be renamed SYS$LIBRARY:OLD_CDDSHR.EXE.

The upgrade installation will also place a new CDD_BUILD_TEMPLATE.COM procedure in SYS$LIBRARY for use with CDD/Repository V5.1.

Note

If you upgrade your repository to CDD/Repository V5.1 after you install Oracle Rdb V7.0, you must install the corrected CDDSHR image again to ensure that the correct CDDSHR images have been made available.

The CDD/Repository upgrade kit determines which version of CDD/Repository is installed and replaces the existing CDDSHR.EXE with the appropriate version of the corrected image.

2.9.2.2 CDD Conversion Procedure

OpenVMS VAX

Oracle Rdb provides RDB$CONVERT_CDD$DATABASE.COM, a command procedure that both corrects the anchor directory ACL and performs the RMU Convert operation. The command procedure is located in SYS$LIBRARY.

Note

You must have SYSPRV enabled before you execute the procedure RDB$CONVERT_CDD$DATABASE.COM because the procedure performs an RMU Convert operation.

Use the procedure RDB$CONVERT_CDD$DATABASE.COM to process the anchor directory and update the ACLs for both the directory and, if available, the repository database.

This procedure accepts one parameter: the name of the anchor directory that contains, or will contain, the repository files. For example:

```
$ @SYS$LIBRARY:DECRDB$CONVERT_CDD$DATABASE [PROJECT.CDD_REP]
```

If many repositories exist on a system, you may want to create a DCL command procedure to locate them, set the Oracle RMU privileges ACL, and convert the databases. Use DCL commands similar to the following:

```
$ LOOP:
$       REP_SPEC = F$SEARCH("[000000...]CDD$DATABASE.RDB")
$       IF REP_SPEC .NES. ""
$       THEN
$           @SYS$LIBRARY:DECRDB$CONVERT_CDD$DATABASE    -
                'F$PARSE(REP_SPEC,,,"DIRECTORY")'
$           GOTO LOOP
$       ENDIF
```

♦

3 Software Errors Fixed

The following sections describe problems with previous versions of the software that are fixed for this release of Oracle Rdb.

This chapter begins with information pertinent to all users. Later sections contain material specifically for users of SQL, RMU, RDO, and RDML.

3.1 Software Errors Fixed That Apply to All Interfaces

This section describes problems that have been fixed for all interfaces.

3.1.1 AIJ Switchover Suspension Prone to DBR-Induced Shutdown

If the AIJ switchover operation cannot complete because there are no available .aij files, the database enters the “AIJ suspended” state. During this state, the DBA can add new .aij files or perform database backups, but all other AIJ-related activities are temporarily suspended until such time as an .aij file becomes available.

During the AIJ suspended period, *any* DBR invocation causes the database to be shut down. This is required because the DBR always writes either a commit or rollback record to the .aij file. Note that even a DBR invoked for a read-only transaction causes the database to be shut down.

The workaround to this problem was to always ensure that adequate .aij files were available for the AIJ switchover operation.

This problem has been corrected in Oracle Rdb V7.0. A new logical name, RDM$BIND_ALS_CREATE_AIJ, and configuration parameter, RDB_BIND_ALS_CREATE_AIJ, have been added. This logical name and configuration parameter indicate whether or not the ALS server is to create an “emergency” AIJ journal if the AIJ switchover operation enters the suspended state. The default value "0" indicates that the ALS should *not* create an .aij file. The value "1" indicates that the ALS should attempt to create an .aij file. On OpenVMS, the logical name must reside in the LNM$SYSTEM_TABLE logical name table.

When the logical name or configuration parameter is set to the value "1", the ALS attempts to create an emergency AIJ journal using the previous .aij file as a template. This means that the emergency AIJ journal is created in the same directory, and with the same allocation, as the current .aij file being switched. If an error occurs, such as inadequate disk space, the database simply enters the “AIJ suspended” state and the DBA must resolve the situation.

Caution

The emergency AIJ journal is *not* a temporary AIJ journal. Do *not* delete it using any means other than through the standard database syntax (SQL or RMU). Deleting the emergency AIJ journal through other means, such as the operating system command line, will cause your database to be shut down.

An emergency AIJ journal is a normal .aij file in all respects. The ALS process is simply created to avoid the AIJ switchover suspension state. DBR invocations due to application process failure during the .aij file creation do not cause database shutdown.

You can specify the location of the emergency AIJ journal (device and directory) by using the RDM$BIND_AIJ_EMERGENCY_DIR logical name or RDB_BIND_AIJ_EMERGENCY_DIR configuration parameter.

OpenVMS VAX OpenVMS Alpha On OpenVMS, if you define this logical in the LNM$SYSTEM_TABLE logical name table, you should only specify the device and directory where the emergency AIJ journal is to be created. If defined, the RDM$BIND_AIJ_EMERGENCY_DIR logical name applies to all databases on the current node. Furthermore, the logical name must not contain any nonsystem concealed logical definitions. ♦

The ALS notifies the DBA through the operator notification facility that an emergency AIJ journal has been created. Furthermore, the RMU Dump Header command identifies any .aij file created by the ALS server process. The Performance Monitor highlights any identified emergency AIJ journal.

When created, the file name of the emergency AIJ journal is "EMERGENCY_XXX" where "XXX" is a series of 16 characters used to create an unique name.

Note

The creation of the emergency AIJ journal is *not* journaled.

There is no way to remove the emergency status of an emergency AIJ journal. However, an emergency AIJ journal is a normal .aij file in all other respects.

3.1.2 Preventing Depletion of AIJ ARB Pool

In previous versions, when a large number of users was attached to a database on a specific node, it was frequently possible to exhaust the AIJ request block (ARB) pool. When this occurred, additional immediate processing was required to make ARBs available. This resulted in a degradation of overall system performance.

This problem was extremely critical during the AIJ switchover operation. Exhausting the ARB pool during AIJ switchover could greatly increase the switchover duration, which greatly impacted all system processing.

There are four ARBs per user, with a maximum of 300 ARBs per node. Each ARB requires 2K of global section space, and a corresponding 2K of user virtual memory (VM).

This problem has been corrected in V7.0.

On OpenVMS, define the logical name RDM$BIND_AIJ_ARB_COUNT in the LNM$SYSTEM_TABLE to define the number of ARBs allocated on the node.

On Digital UNIX, use the configuration parameter RDB_BIND_AIJ_ARB_COUNT. There is no maximum value.

The default value is four times the maximum number of users minimized to 300.

3.1.3 Process Starvation and Hang During AIJ Switchover

In previous versions, when using the AIJ Log Server (ALS) process to manage the AIJ group commit operations, it was possible during extremely highly CPU-intensive operations that an .aij file switchover operation would result in 100% saturation of all processors and the database would hang.

This problem only occurred when using the ALS process, and only when the ARB pool had been exhausted. Under normal operating circumstances, this situation was very rare.

The problem was caused by application processes trying to find space to flush their AIJ records, when there was no space. The ALS process was trying to flush the AIJ information to the .aij file, but was waiting for the AIJ switchover operation to complete. The AIJ switchover operation could not complete because not all processes could get enough CPU time to acknowledge the AIJ switchover protocol.

The workaround was to increase the size of the .aij file so that AIJ switchover did not occur during the busiest application processing periods. In addition, the RMU Set After_Journal command with the Switch qualifier did not exhibit the described behavior and could have been used as a workaround.

This problem has been corrected in V7.0. Application processes do not allow the ALS process to complete the AIJ switchover operation by releasing their CPU time-slice when no more AIJ ARBs are available.

3.1.4 After-Image Journal File Switchover Race Condition Corrected

In previous versions, during an AIJ switchover operation, it was possible for the last commit transaction sequence number (TSN) stored in the new .aij file open record to not be the last committed transaction TSN from the previous .aij file. This was caused by the asynchronous nature of the .aij group commit mechanism.

This problem has been corrected in V7.0. The race condition that caused the problem has been corrected; the last commit TSN of the new .aij file open record accurately reflects the last committed transaction's TSN from the old .aij file.

3.1.5 Failure to Open After-Image Journal No Longer Causes Locking Problems

In previous versions, it was possible that a failure to open the .aij file would not correctly clean up its locks, which led to subsequent locking problems. This problem typically occurred during transaction start, or following an AIJ switchover operation.

The cause of the failure to open the .aij file was usually resource related, typically PGFLQUOTA or BYTLIM quota.

There was no workaround to the dangling lock problem. The affected process needed to disconnect from the database to resolve the problem.

This problem has been corrected in V7.0. Any failure during the opening of an .aij file now correctly cleans up affected resources.

3.1.6 DDL Operations on After-Image Journal Files No Longer Deadlock with AIJ Switchover

In previous versions, it was sometimes possible for an AIJ switchover operation to become deadlocked while performing data definition language (DDL) modifications to the after-image journal.

The workaround was to not perform DDL operations on the after-image journal if there was the possibility of an imminent AIJ switchover operation about to occur.

This problem has been corrected in V7.0.

3.1.7 AIJ Inaccessible After Node or Cluster Failure or When the Database Is Stopped with Abort=Delprc Qualifier

In previous versions, the AIJ Log Server terminated abnormally and the current AIJ file became inaccessible under the following database conditions:

- You used the SQL statement ALTER DATABASE with the NUMBER OF CLUSTER NODES IS clause set to 1.
- An AIJ cache file on an electronic disk and the AIJ log server (ALS) process were enabled.
- The database was stopped with Abort=Delprc qualifier with an active transaction and, subsequently, the database was reopened.

In the case of cluster failure or node failure, because the database needed recovery, and the monitor ID for the dead node was 1, the monitor assigned the database a new ID 2 for recovery reasons, *even* if it was the same monitor as before. However, since the database was created with NODES=1, there was no room in the ACE file when the monitor ID exceeded the maximum number of nodes. The result was an inaccessible .aij file.

This problem involved all interfaces.

The workaround to this problem was to change the number of database nodes to a minimum of 2 or to disable the AIJ cache file on an electronic disk.

This problem has been corrected in V7.0. Now, when the ACE file is created, its size is based on the maximum number of nodes in the database. When the ACE file is accessed, the single file is partitioned among the various nodes based on the monitor ID (the monitor ID is normally in the range 1:n where "n" is the number of nodes in the database).

3.1.8 User Processes Do Not Hibernate on AIJ Submission

In previous versions, it was possible, on rare occassions, for user processes to stall for long periods when writing to the AIJ. The stall messages screen in the Performance Monitor indicated those processes as "Hibernating on AIJ submission".

The problem was caused by a small race condition between the AIJ Log Server (ALS) process determining whether it needed to hibernate, and application processes determining whether to wake up a hibernating ALS process.

The workaround was to stop using the ALS.

This problem has been corrected in V7.0.

3.1.9 Performance No Longer Degrades in Dynamic OR Optimization

OpenVMS VAX

In previous versions, some queries using the OR predicate, for which Oracle Rdb uses the dynamic OR optimization technique, sometimes experienced performance slowdown. The following query, where table T1 has an index on column F1, uses the dynamic OR optimization technique (the debug flag is defined as S):

```
SELECT F1, F2 FROM T1 WHERE F1 = :F1A OR F1 = :F1B
Conjunct       Get     Retrieval by index of relation T1
  Index name  I1 [1:1...]2
```

The slowdown happened if a query, such as the previous one, was executed multiple times and one of the executions was such that the values supplied for the OR predicate collapsed into one range. All subsequent executions could have suffered from performance degradation, as Oracle Rdb scanned the complete range of index keys between the two supplied values.

This problem has been corrected in V7.0. ♦

3.1.10 Recovery and Fast Commit No Longer Results in Database Corruption

In previous versions, there was a rare failure scenario where databases using the fast commit feature could have lost updates to the database. The problem occurred when multiple processes failed and had to be recovered by database recovery (DBR) processes. For example:

1. Process 1 updated a database page and flushed the page to disk. No commit was done.
2. Process 2 updated the same page and committed, but did not checkpoint; thus the page update was not flushed to disk.
3. Process 1 failed and a DBR process began to recover the process. Because the process did not commit, its updates were rolled back (undone) by the DBR process. The DBR process incorrectly incremented the page sequence number on the page, making the page appear to have contained updates done by process 2 even though the updates were not actually on the page.
4. Process 2 failed and a DBR process began. In this case, because the process had committed without a checkpoint, its updates needed to be redone. The DBR process examined the page, found that the page sequence number on the page indicated that the updates from process 2 were on the page, and incorrectly determined that those lines did not need to redone. The page updates done by process 2 were lost.

This problem has been corrected in V7.0.

3.1.11 Recovery Process No Longer Hangs When Using Global Buffers

OpenVMS VAX OpenVMS Alpha

In previous versions, when using global buffers, if a user process was abnormally terminated with the DCL STOP/ID command (or DELETE/ENTRY for a batch job), the DBR process could have become deadlocked with another user process. The only solution to this problem was to stop the DBR process, which resulted in the database being shut down.

The deadlock condition was due to a user process holding a global section latch (in this context, latch is a type of lock within the global section) while it was waiting for a freeze lock. The DBR process was waiting for the user process's latch, causing the freeze.

As a workaround, when making very heavy use of global buffers, you would not have abnormally terminated user processes with the STOP/ID command or, in the case of batch jobs, DELETE/ENTRY.

This problem has been corrected in V7.0. The DBR process now correctly identifies and resolves the deadlock condition. ♦

3.1.12 DBR No Longer Fails During REDO When Fast Commit Is Enabled

In previous versions, when the AIJ Fast Commit feature was enabled, it was possible for the (DBR) process to fail during the Redo phase because insufficient space existed on the affected data page.

A dump of the affected page showed that it had some locked free space reserved by a process that wrote that page several days before. The RMU Dump Users command also revealed that this TID was still active. Since this was an unexpected problem, the DBR process bugchecked.

A scenario such as the following could cause this problem:

1. A process, P1 with TID N, had deleted all rows from page STAREA:PNO several days ago.
2. Another process, P2, stored 2 new rows on that page and committed the data. The page was not written out (because fast commit was enabled).
3. Process P3 attached to the database, and was assigned TID N again.
4. Process P1 failed due to an application error.
5. The DBR Redo process for P1 failed because all space on page STAREA:PNO appeared to be reserved.

The only workaround was to disable the AIJ Fast Commit feature.

This problem has been corrected in V7.0. The DBR process now correctly analyzes the data page with respect to active processes.

3.1.13 DBR No Longer Rolls Back Committed Transaction Data

In previous versions, when you enabled the commit-to-journal option, there was a scenario whereby the DBR process sometimes rolled back the last transaction to have written .ruj records for a failed process. The rollback occured when the failed process had a committed read-only transaction perform a checkpoint operation subsequent to the last database modification transaction.

Only the information in the .ruj file was rolled back. If the committed transaction did not write an .ruj record for every database modification, the transaction was partially rolled back. This sometimes resulted in a corrupted database.

The problem was caused by the introduction of read-only transactions, which caused the commit-to-journal option to be dynamically disabled.

This problem only occurred when the commit-to-journal option was enabled and read-only transactions were frequently utilized. Furthermore, this problem only occurred when a read-only transaction performed a checkpoint operation subsequent to committing a read/write transaction, and then, subsequent to the checkpoint, the process failed abnormally.

The problem did not occur if read-only transactions were rolled back instead of committed.

You could identify the problem by using the RMU Dump Header command to examine the .aij file. A rollback record with a TSN less than the last commit record of the same process was an indication that the problem had occurred.

The workaround was to disable the .aij commit-to-journal option or to use a read/write transaction.

This problem has been corrected in V7.0.

3.1.14 DBR Now Validates Checkpoint During REDO

OpenVMS VAX OpenVMS Alpha

In previous versions, when using concealed logical names for the .aij filename, it was possible for the (DBR) process to access the incorrect .aij file. This occurred because the DBR process is invoked by the database monitor, typically through the SYSTEM account.

The result was that the DBR process attempted to perform transaction REDO using an .aij file that did not directly correspond to the database that the failed process was accessing. This condition was not detected.

The result of this problem was that the DBR process *might* have rolled back the last committed transaction for the failed process.

The workaround was to not use concealed logical names.

This problem has been corrected in V7.0. The DBR process will bugcheck, and the database will be immediately shutdown if the proper .aij file cannot be accessed. ♦

3.1.15 PAGE TRANSFER VIA MEMORY and Fast Incremental Backup No Longer Cause PIO$MARK_SNUB Bugcheck

In previous versions, when the PAGE TRANSFER VIA MEMORY and fast incremental backup features were both enabled, it was possible to produce a bugcheck dump when a storage area readied to concurrent read access fetched a modified page from another process.

The workaround was to disable either the PAGE TRANSFER VIA MEMORY feature or the fast incremental backup feature.

This problem has been corrected in Oracle Rdb V7.0.

3.1.16 Bugchecks at PIOFETCH$WITHIN_DB + 0784 Eliminated

OpenVMS Alpha

In previous versions, Oracle Rdb sometimes bugchecked in PIOFETCH$WITHIN_DB + 0784 because a register was incorrectly sign extended from 32 bits to 64 bits.

The following extract from an Oracle Rdb dump file shows an example call trace of this problem:

```
***** Exception at 02B1B584 : PIOFETCH$WITHIN_DB + 000007C4
%COSI-F-BUGCHECK, internal consistency failure
Saved PC = 02B18C2C : PIO$FETCH_RET + 000002BC
Saved PC = 02B17EE4 : PIO$FETCH + 0000036C
```

Usually, the operation was successful if you tried it again.

This problem has been corrected in V7.0. ♦

3.1.17 Bugcheck at Transaction Commit Is Now Fixed

In previous versions, certain conditions caused committing the current transaction to produce a bugcheck dump. This occurred under a series of events related to record lock acquisition and conflicts, which resulted in the bugcheck at commit time.

This problem has been corrected in V7.0.

3.1.18 Now Can Create a Database with Lock Partitioning and Global Buffers

OpenVMS VAX OpenVMS Alpha

In previous versions, when you created a database and specified both GLOBAL BUFFERS ENABLED and LOCK PARTITIONING ENABLED, Oracle Rdb bugchecked with a COSI-F-ACCVIO exception.

The following example shows how this problem could have occurred:

```
SQL> CREATE DATABASE FILENAME FOO
cont> GLOBAL BUFFERS ENABLED
cont> LOCK PARTITIONING ENABLED;
%SQL-I-BUGCHKDMP, generating bugcheck dump file DSK$:[DIR]SQLBUGCHK.DMP;
%COSI-F-ACCVIO, access violation
```

A workaround was to not specify LOCK PARTITIONING ENABLED when creating a database. Once the database had been created, you could enable this feature using the ALTER DATABASE command.

This problem has been corrected in V7.0. ♦

3.1.19 Oracle Rdb No Longer Fails on OpenVMS Alpha V7.0

OpenVMS Alpha

Previous versions of Oracle Rdb running on OpenVMS Alpha V7.0 generated bugchecks within the OpenVMS exception handling code. This occurred when certain Oracle Rdb exceptions were raised. When raising the exception, the OpenVMS exception handling system code failed with an access violation (accvio) exception.

The following extract from an Oracle Rdb bugcheck dump file shows the exception within the OpenVMS exception handling routine in S0 (system) address space after an exception was signaled in the Oracle Rdb LCK$LOCK routine:

```
 Saved PC = 01065774 : KOD$BUGCHECK_DUMP + 00001014
 Saved PC = 00DA3C6C : RDMS$$TOP_DSDI_CLEANUP + 0000021C
 Saved PC = 00DA3634 : RDMS$$TOP_DSDI_HNDLR + 0000017C
 Saved PC = 80007164 : S0 address
 Saved PC = 8C4D059C : S0 address
 ***** Exception at 8C4CE670 : S0 address
 %SYSTEM-F-ACCVIO, access violation, reason mask=00,
 virtual address=00000000011D8F28, PC=FFFFFFFF8C4CE670, PS=00000009
 Saved PC = 00F6A7A0 : LCK$LOCK + 00000908
 Saved PC = 00E84344 : RDMS$$RDMSCHEMA_LOAD_TABLE + 00000D0C
 Saved PC = 00CACADC : RDMS$$RETURN_SYMBOL + 0000008C
```

This problem was due to the way that the Oracle Rdb RDMSHRP image was linked. This image has a static copy of LIB$SIGNAL from the object library STARLET.OLB (from OpenVMS V1.5 or V6.1, depending on the Oracle Rdb version) linked with it. This older LIB$SIGNAL code is not compatible with the new exception code in OpenVMS Alpha V7.0, because of changes in the OpenVMS mechanism array format. Therefore, Oracle Rdb caused the access violation exception in the OpenVMS system code.

This problem has been corrected in V7.0. The RDMSHRP image is linked against the LIBRTL.EXE image, which allows the corrent LIB$SIGNAL routine to be used for the OpenVMS version being run. ♦

3.1.20 Multiple Connections No Longer Cause Missing Oracle Trace Data

OpenVMS VAX OpenVMS Alpha

In previous versions, when collecting Oracle Trace information and using connections in Oracle Rdb, certain information was not being passed from Oracle Rdb to Oracle Trace. This could have caused Oracle Trace to produce incorrect results.

The following example shows an SQL session that does not use connections. Oracle Trace events collected for this session would not have included all required data for connection CON_2.

```
SQL> ATTACH 'FILE MF_PERSONNEL';
SQL>
SQL> SELECT * FROM SALARY_HISTORY LIMIT TO 1 ROWS;
SQL> ROLLBACK;
SQL>
SQL> CONNECT AS 'CON_1';
SQL> CONNECT AS 'CON_2';
SQL>
SQL> SET CONNECT 'CON_1';
SQL> SELECT * FROM COLLEGES LIMIT TO 2 ROWS;
SQL> ROLLBACK;
SQL>
SQL> SET CONNECT 'CON_2';
SQL> SELECT * FROM DEPARTMENTS LIMIT TO 3 ROWS;
SQL> ROLLBACK;
```

There were no workarounds to this problem.

This problem has been corrected in V7.0. ♦

3.1.21 Nominal Record Length Now Stored in AIP for UNIQUE Indexes

In previous versions, when a user created an index and did not specify a NODE SIZE parameter for the index, Oracle Rdb created the logical area, with a default record length of 215, which it stored in the area inventory page (AIP) and used for searching for space for creating new index nodes. Some applications experienced prolonged waits while inserting or updating records due to an excessive number of pages being checked before creating a new index node.

The recommended method to alter this behavior and improve performance is to determine proper threshhold parameters for the storage areas or to explicitly specify a NODE SIZE parameter when creating the index.

In V7.0, the RECORD LENGTH parameter used to create the logical areas for UNIQUE indexes is now the same as that used for the default index node size, because there is no need to allow for space for duplicates nodes in these logical areas.

However, there is a potential side-effect. If two or more sorted indexes are created in the default storage area, the RECORD LENGTH value stored in the AIP depends on whether the first index created was UNIQUE or NOT UNIQUE.

For related information, see Section 3.1.22.

3.1.22 Node Size Calculation for Unique Sorted Indexes

When a unique sorted index is defined in V7.0, the default node size is calculated differently than in previous versions.

In previous versions, if a unique sorted index was defined without specifying the NODE SIZE clause, the AIP length for the logical area was estimated as 215 bytes. Often, this caused pages to be fetched from disk and then discarded because the actual node would not fit on the page.

In V7.0, this estimation has been changed, as follows:

- The smallest node size for a sorted index can be calculated using the following formula:
  - keylength = sum of all column lengths, plus 1 null byte per column (the keylength may not exceed 255 bytes)
  - overhead = 11 (or 18 for sorted ranked indexes)

    The extra overhead for sorted ranked indexes is used to store cardinality estimates to assist the optimizer.
  - minimum_length = 3 * (keylength + overhead) + 32
- If no NODE SIZE is specified, Oracle Rdb defaults to 430 bytes unless this is greater than the minimum_length shown in the previous calculation, in which case Oracle Rdb defaults to 860 bytes.
- If a NODE SIZE is specified, it must be greater than the minimum_length shown in the calculation.
- The maximum NODE SIZE that can be specified for a sorted index is 32767 bytes.

3.1.23 Performance Enhancement for Storage Maps and Mapped Indexes

In previous versions, some large applications with very many partitions for some tables noted a performance degradation. The degradation was in the form of excessive amounts of virtual memory and CPU usage.

The Oracle Rdb partitioning scheme permits great flexibility for spreading data across storage devices. Unfortunately, this very flexibility turns into a shortcoming for databases with tables and indexes spread across hundreds of devices or files. For these very large databases, a simpler scheme was developed which had much more desirable performance characteristics. Beginning with V6.0, whenever a table or index was partitioned using only an unscaled, integer column (INTEGER or SMALLINT), Oracle Rdb automatically implemented a simpler, better performing algorithm.

Beginning with V7.0, Oracle Rdb extends the use of this algorithm to almost any partitioning scheme that only relies upon a single key for its decision. Note that the partitioning criteria must match the data type of the column used for partitioning. The exception to the use of this algorithm is when the partitioning criteria uses columns that are scaled integers.

3.1.24 Memory Leak Plugged for Insert with Storage Maps

In previous versions, if the user executed an INSERT statement for a table with storage maps, the memory requirement for the user's process tended to grow. It would take quite some time for the user to notice the problem and most likely this would only happen in a nonprecompiled environment.

This problem has been corrected in V7.0.

3.1.25 Update on Rows with Many Missing Values

In previous versions, some unusual update operations encountered bugchecks. The columns in the tables which were updated used the RDO MISSING VALUE construct. Manifestations of the error included either bugchecks or unexpected results. The bugcheck was of the form `Exception at XXXXXXX : STR$COPY_R_R8 + XXXXXXX"`

One workaround was to update fewer columns in the update statement because the problem occurred only when five or more columns were updated in one statement.

This problem has been corrected in V7.0.

3.1.26 Long Records on Alpha Platforms No Longer Cause Problems

OpenVMS Alpha Digital UNIX

In previous versions, if a user created a table with records longer than 32767 bytes and altered the table, a bug appeared on Alpha platforms. The most obvious manifestation of the bug was a bugcheck at RDMS$$OTS_MOVE+8. Less obvious manifestations were loss of null bits in the current versions of the record, or data corruption in the executive — which may or may not have made itself known. If such records were not updated on Alpha platforms, future versions of Oracle Rdb read the records correctly. If they were updated, Oracle Rdb recommended reviewing the data for possible loss of NULL information.

This problem has been corrected in V7.0. ♦

3.1.27 Record Compression on Alpha Platforms

OpenVMS Alpha Digital UNIX

In previous versions, certain records may have been compressed incorrectly on Alpha platforms. Only records that ended with 129 or 130 zeroes—or 128*n+(129 or 130) and had a multiple of 8 columns in the record, none of which were null, were compressed incorrectly. The string of zeroes had to start at a particular offset in the record, such that if the earlier conditions were met, there was only a 1 in 8 chance of encountering the bug.

There were also scenarios for bit patterns other than zeroes that also required particular patterns for the nulls in the record. For example, with spaces as the last string of bytes, every second, tenth, eighteenth, and so forth, column in the record would have to be null to encounter this bug. Retrieval of such a record would either result in a bugcheck or erroneous data.

The following shows an example of the problem:

```
CREATE DATABASE FILENAME FOO;
CREATE DOMAIN Q BIGINT;
CREATE TABLE BAR (X1 CHAR(1),X2 CHAR(1), X3 CHAR(1), X4 CHAR(1),
                  X5 CHAR(1), X6 CHAR(1), X7 CHAR(1), X8 CHAR(4),
                  F1 Q, F2 Q, F3 Q, F4 Q, F5 Q, F6 Q, F7 Q, F8 Q,
                  F9 Q, F10 Q, F11 Q, F12 Q, F13 Q, F14 Q, F15 Q, F16 Q);
INSERT IN BAR VALUES (' ',' ',' ',' ',' ',' ',' ',' ',
               7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0);
```

```
SELECT f1 FROM BAR;
               F1
          1638407
1 row selected
```

Workarounds included not using compression or adding a column to the record.

This problem has been corrected in V7.0. ♦

3.1.28 Database No Longer Hangs If Process Holding Logical Area Lock Does Not Release Lock

In previous versions, Oracle Rdb database applications would sometimes hang waiting for a process that did not have a database transaction active to release a logical area lock. During the commit process there was a small window of time where Oracle Rdb could receive a blocking AST (BlAST) from another process for a logical area lock but the lock would not be released. The process would continue to hold the lock until a new transaction was started and the process received another blocking AST.

This problem has been corrected in V7.0.

3.1.29 Excessive Root File I/O

In previous versions, it was possible for Oracle Rdb to excessively read the database root (.rdb) file after a database process had an abnormal termination or after a cluster node accessing the database failed. The problem was caused by locking data structures becoming invalidated due to the abnormal failure and not being properly validated again.

This problem has been corrected in V7.0.

3.1.30 Applications No Longer Hang When Executive Mode ASTs Are Disabled

OpenVMS VAX OpenVMS Alpha

In previous versions, in rare situations, application processes using Oracle Rdb would hang when executive mode ASTs were disabled. Generally, if a process is hung in this state, it is not possible to kill this process using the DCL STOP command. This condition could have resulted in a hung database requiring a system reboot to clear.

This condition was sometimes caused by an exception (for example, an access violation) occuring in an Oracle Rdb executive mode AST routine, such as a lock blocking AST routine. Once an exception occurred, the OpenVMS system scanned the call frames looking for condition handlers to call. If the condition handler that was called attempted any RMS I/O (to write out an error message, for example), the RMS I/O could not complete because the process was at executive mode AST level. RMS routines are not allowed to be called at that level. Oracle Rdb did not enable its own condition handler in the executive mode AST routines.

This problem has been corrected in V7.0. Oracle Rdb now enables a condition handler in the executive mode lock blocking AST routines. This handler correctly lowers the AST level before starting an Oracle Rdb bugcheck dump. ♦

3.1.31 CREATE INDEX with SIZE IS Clause No Longer Returns Incorrect Results

In previous versions, if the user created an index for a CHAR column with a SIZE IS clause which exactly matched the size of the column, some queries returned incorrect results.

The following example shows the previous, incorrect behavior. Note that the rows for Tom Robinson should not have been returned for this query.

```
SQL> DROP INDEX emp_employee_id;
SQL> DROP INDEX emp_last_name;
SQL> CREATE INDEX fn on employees (first_name size is 10, city);
SQL> CREATE INDEX fx on employees (first_name, last_name, address_data_1);
SQL> SELECT * FROM employees
cont>    where (first_name = 'Tom' and last_name = 'S' and employee_id >'0')
cont>       or (first_name = 'Tom' and last_name > 'S')
cont>       or (first_name >'Tom       ');
 EMPLOYEE_ID   LAST_NAME       FIRST_NAME   MIDDLE_INITIAL
   ADDRESS_DATA_1             ADDRESS_DATA_2         CITY
      STATE   POSTAL_CODE   SEX    BIRTHDAY      STATUS_CODE
 00229        Robinson        Tom          A
   134 Lantern Lane                                   Fremont
      NH      03044         M      24-Feb-1934   1

 00183        Nash            Walter       V
   197 Lantern Lane                                   Fremont
      NH      03044         M      19-Jan-1925   1

 00205        Bartlett        Wes          NULL
   28 Page Hill Rd.                                   Meadows
      NH      03587         M       16-Apr-1950   1

3 rows selected
```

The workaround was to omit unneccessary SIZE IS clauses on the CREATE INDEX statement or use a smaller value.

This problem has been corrected in V7.0.

3.1.32 LOCK_CONFLICT Error on Multiple Databases No Longer Leaves Transaction Active

In previous versions, if you started a transaction on multiple databases and a LOCK_CONFLICT error occurred on one of the databases, Oracle Rdb did not roll back any transaction which may have been started on the other databases. However, because the user's transaction handle was cleared, there was no way to explicitly roll back the started transaction. Any subsequent attempt to start another transaction resulted in an EXCESS_TRANS error.

There was no workaround, other than starting the transactions on the databases one at a time.

This problem has been corrected in V7.0.

3.1.33 System Metadata Index Corruption Fixed

In previous versions, indexes on the system tables could have become inconsistent if multiple users were performing database definition operations at the same time. This could have led to a variety of errors. For example, Oracle Rdb sometimes became unable to locate tables that were defined in the database. Various bugchecks could occur due to inconsistencies in the system metadata. For example:

```
***** Exception at 00D06590 : PSII$REMOVE_BOTTOM + 000005F0
%COSI-F-BUGCHECK, internal consistency failure

***** Exception at 0039E529 : RDMS$$INSERT_SYMBOL + 000000EF
%COSI-F-BUGCHECK, internal consistency failure
```

The errors occurred because Oracle Rdb released locks on system metadata index nodes prior to the end of the transaction. Releasing the locks made it possible for there to be buried updates on index data which could result in corrupted indexes.

Index nodes that were locked using the ISOLATION LEVEL READ COMMITTED clause could have locks released prior to the end of a transaction. System metadata indexes are locked using this feature. Application code that used the feature may also have encountered index corruption in indexes used by user-defined tables.

This problem has been corrected in V7.0.

3.1.34 Checksum Errors on Alpha Processors Fixed

OpenVMS Alpha Digital UNIX

In V6.1, in rare circumstances, it was possible for Oracle Rdb to store pages in the database with incorrectly calculated page checksums. Attempts to access those pages resulted in checksum errors. The checksum was always off by one. For example, Oracle Rdb may have reported an error such as the following:

```
%RDMS-F-CANTREADDBS, error reading pages 2:14906-14906
-RDMS-F-CHECKSUM, checksum error - computed F7CE0831, page contained F7CE0830
```

This problem existed only in Oracle Rdb V6.1 on Alpha platforms. Checksum errors that do not exhibit a difference of one are not caused by this problem. Most checksum errors are caused by I/O subsystem problems. If the symptoms do not exactly match the scenario described, the problem is likely to be due to errors in the I/O subsystem, such as disks, controllers, or software drivers.)

This problem has been corrected in V7.0. ♦

3.1.35 Bugchecks at PIOAPF$AST + 78 Fixed

In previous versions, Oracle Rdb sometimes bugchecked in the routine PIOAPF$AST with an ACCVIO error. This problem occurred only in processes that were running as a batch job; it was due to conflicts with the SYS$FLUSH RMS call used by DCL to flush the batch log file.

This problem sometimes caused random data structures to get corrupted in shared memory (database global section) and thus other database processes could have failed with various bugchecks at the same time.

To prevent or reduce the incidence of the problem, you could have done one of the following:

- Increase the interval for batch log flushing using the DCL SET OUTPUT_ RATE command.
- Disable asynchronous prefetch via the RDM$BIND_APF_DISABLED logical.

This problem has been corrected in V7.0.

3.1.36 Detach Failure No Longer Returns Invalid Request Handle

Some high-throughput applications, such as transaction processing applications, have found it desirable to use some specific capabilities of Oracle Rdb: the RDM$BIND_LOCK_TIMEOUT_INTERVAL logical name, frequent disconnects and attaches within the same program execution, and several processes updating the database with this behavior. In previous versions, such applications occasionally failed on the disconnect because of a timeout while updating the cardinalities in the system tables. The net effect of this failure was that subsequent queries might have failed with the error RDB$_BAD_REQ_HANDLE.

This error has been corrected in V7.0. The disconnect proceeds and ignores the error returned while attempting to update the approximate cardinalities in the system tables. Note that, as a result of this correction, the cardinalities are not necessarily updated.

3.1.37 Error Details No Longer Lost from Remote Prefetch Operations

In previous versions, if you fetched rows or list data from a remote database and an error occurred after the first row or segment, Oracle Rdb did not report all the details on what went wrong.

For example, if you set the query limit with the SQL statement SET QUERY LIMIT ROWS 3 and then executed an SQL SELECT * statement on a remote database, the error message that appeared after the third row reported that a quota has been exceeded, but not which quota.

The workaround was to disable prefetch by placing the following two lines in the RDB$CLIENT_DEFAULTS.DAT (OpenVMS) or .dbsrc (Digital UNIX) file:

```
SQL_RCV_PREFETCH_ROWS 0
SQL_SGS_PREFETCH_ROWS 0
```

This problem has been corrected in V7.0.

3.1.38 Error Details No Longer Lost from Remote Databases

In previous versions, when transmitting status codes from a remote database, Oracle Rdb garbled the information in the message vector if it encountered a code from any facility other than "RDB" and the message code image for that facility was not linked into the client-side program.

For example, an Oracle Rally application could not fetch the name of a failing trigger in a remote database because that particular status code belongs to "RDMS" instead of "RDB", and Rally does not link in the "RDMS" message codes.

The workaround was to use the DCL SET MESSAGE command before running the application program, to cause the missing message file to be available.

This problem has been corrected in V7.0.

3.1.39 Monitor No Longer Hangs After Certain Period of Activity

In previous releases, it was possible for the database monitor process to suddenly stop processing user attaches and appear to be hung. The monitor process had to be manually killed and re-started to make the system operational.

This situation could have been determined by analyzing the system using the VMS SDA utility and examining the open channels (using the DCL SHOW PROC /CHANNEL command). The mailbox channels (typically 2 of them, starting with MBAn) ordinarily both have a status of "busy". If the mailboxes did not have a status of busy, you encountered this problem.

There was no workaround to this problem. Issuing an RMU Open command when the database was already open or an RMU Monitor Reopen_Log command made the situation more likely to occur.

This problem has been corrected in V7.0.

3.1.40 Read-Only Transactions No Longer Fail with Deadlocks on SNAPSHOT CURSOR 0

In previous versions, when you enabled the fast commit option and set snapshots to enabled deferred, the SET TRANSACTION READ ONLY statement sometimes failed with a deadlock on resource "SNAPSHOT CURSOR 0".

The resource SNAPSHOT CURSOR 0 is used to control the transition between inactive and active snapshots, when snapshots are in deferred mode.

The problem occurred if the process starting the read-only transaction retained locks on pages updated in a prior read/write transaction (as is the case when the fast commit option is used). If a concurrent read/write process wanted to update one of those pages, the deadlock occurred.

The following timeline illustrates the circumstances leading to the deadlock (assuming a simple database with one table).

```
 Process A                      Process B

ATTACH 'DA FI DEMO';
                               ATTACH 'DA FI DEMO';
                               SELECT * FROM T1;
SET TRANS READ WRITE;
                               SET TRANS READ ONLY;
                               (stalls: waiting for snapshot cursor 0)
DELETE FROM T1;
 (stalls: waits for the first page of
 table T1. Processes are in deadlock)

                               (transaction fails with %RDB-E-DEADLOCK)
(operation proceeds and completes)
```

This problem has been corrected in V7.0. The read-only transaction now releases outstanding page locks before it starts stalling for the snapshot cursor 0 lock.

3.1.41 Attached Inactive Processes Now Perform Global Checkpoint Operations

Beginning with V6.0A ECO 2 (V6.0-12), attached but inactive processes did not always respond to global checkpoint operations. This rendered the AIJ backup utilities, both manual and automatic, unable to backup the .aij file to which the inactive processes referred.

The workaround was to detach inactive processes from the database.

This problem has been corrected in V7.0. Attached but inactive processes now correctly respond to global checkpoint operations.

3.1.42 Undetected Global Checkpoint Deadlock Corrected

With previous versions, it was possible for the RMU Checkpoint command to result in an undetected deadlock situation.

If an RMU Checkpoint command was issued while a process was waiting for a resource used by another process, the global checkpoint operation hung. If at that time, the second process decided to commit or rollback, the first process, the second process, and the RMU Checkpoint command hung in a deadlock situation that was not detected by the Lock Manager.

This was a deadlock situation where User 2 was waiting for the page lock that User 1 had and User 1 was waiting for the 'global checkpoint' lock that User 2 had. User 3 was waiting for both to acquire the global checkpoint so the operation could be considered finished. However, User 2 acquired the global checkpoint lock in a way that told the lock manager not to signal a deadlock for that lock, therefore the deadlock was not detected.

You may think that because a process cannot checkpoint in the middle of a transaction, making User 1 execute a COMMIT or ROLLBACK would clear the situation. However, this was not the case. If User 1 committed or rolled back, that process hung as well, for the previously mentioned reason.

The following example shows how to reproduce this problem using the mf_ personnel database:

```
USER 1  ( PID: 2020232D )
======
Aztech RTA10:> SQL$
SQL> ATTACH 'FILENAME TEST';
SQL> UPDATE SALARY_HISTORY SET SALARY_AMOUNT=30000 WHERE EMPLOYEE_ID='12345';
1 row updated
SQL>

USER 2  ( PID: 202022FD )
======
Aztech RTA4:> SQL$
SQL> ATTACH 'FILENAME TEST';
SQL> UPDATE SALARY_HISTORY SET SALARY_AMOUNT=30000 WHERE EMPLOYEE_ID='00100';

(Process hangs)
```

At this point, the Performance Monitor looked as expected:

```
Node: AZTECH         Oracle Rdb V6.0-11 Performance Monitor   14-JUN-1995 08:46:59
Rate: 3.00 Seconds              Stall Messages           Elapsed: 00:16:30.31
Page: 1 of 1      DISK$_1:[JOE_USER.DIRECT.RDB60]TEST.RDB;1      Mode: Online

Process.ID Since......   Stall.reason............................ Lock.ID.
202022FD:1 08:43:28.52 - waiting for logical area 49 (PR)         28000DB5
```

An RMU Show Locks command with the Mode=Blocking qualifier showed the following:

```
================================================================================
SHOW LOCKS/BLOCKING Information
================================================================================

--------------------------------------------------------------------------------
Resource: logical area 49

          ProcessID Process Name       Lock ID   System ID Requested Granted
          --------- ---------------    --------- --------- --------- -------
Waiting:  202022FD  njl @ RTA4.....    28000DB5  00010001  PR        NL
Blocker:  2020232D  njl @ RTA10....    02001E17  00010001  PR        PW
```

Now, if User 3 issued an RMU Checkpoint command, that process hung as well:

```
USER 3  ( PID: 20202304 )
======

$ RMU/CHECKPOINT TEST

 (process hangs)
```

At this point, the Performance Monitor and RMU Show Locks output showed the following:

```
Node: AZTECH         Oracle Rdb V6.0-11 Performance Monitor   14-JUN-1995 08:55:26
Rate: 3.00 Seconds              Stall Messages            Elapsed: 00:24:57.31
Page: 1 of 1      DISK$_1:[JOE_USER.DIRECT.RDB60]TEST.RDB;1      Mode: Online

Process.ID Since......   Stall.reason............................ Lock.ID.
202022FD:1 08:43:28.52 - waiting for logical area 49 (PR)         28000DB5
20202304:1u08:55:18.92 - waiting for global checkpoint (EX)       340025EA
2020232D:1 08:55:19.23 - waiting for global checkpoint (CR)       0700161A

================================================================================
SHOW LOCKS/BLOCKING Information
================================================================================

--------------------------------------------------------------------------------
Resource: logical area 49

          ProcessID Process Name       Lock ID   System ID Requested Granted
          --------- ---------------    --------- --------- --------- -------
Waiting:  202022FD  njl @ RTA4.....    28000DB5  00010001  PR        NL
Blocker:  2020232D  njl @ RTA10....    02001E17  00010001  PR        PW

--------------------------------------------------------------------------------
Resource: global checkpoint

          ProcessID Process Name       Lock ID   System ID Requested Granted
          --------- ---------------    --------- --------- --------- -------
Waiting:  2020232D  njl @ RTA10....    0700161A  00010001  CR        NL
Blocker:  20202304  njl @ RTA12....    340025EA  00010001  EX        NL
Blocker:  202022FD  njl @ RTA4.....    01002CA6  00010001  CR        CR

--------------------------------------------------------------------------------
Resource: global checkpoint

          ProcessID Process Name       Lock ID   System ID Requested Granted
          --------- ---------------    --------- --------- --------- -------
Waiting:  20202304  njl @ RTA12....    340025EA  00010001  EX        NL
Blocker:  202022FD  njl @ RTA4.....    01002CA6  00010001  CR        CR
```

The workaround was to use the RMU Checkpoint command with the Nowait qualifier.

This problem has been corrected in V7.0. The RMU Checkpoint command has been corrected to perform the global checkpoint operation in a manner that does not cause an undetected deadlock situation to occur.

3.1.43 Null Fields Now Detected from Versioned Tables

OpenVMS Alpha

In previous versions on OpenVMS Alpha systems, Oracle Rdb sometimes incorrectly handled null columns in rows from older versions of a table. Null columns could have returned the default column value rather than the null indicator as expected. The same operation worked correctly on an OpenVMS VAX system.

In the following example, rows with a null STOCK_QUANTITY exist, but Oracle Rdb returns a count of zero because the null column was not properly detected:

```
SQL> SELECT COUNT(*) FROM STOCK WHERE STOCK_QUANTITY IS NULL;

          0
1 row selected
```

The workaround was to use an OpenVMS VAX system to modify all rows in the table by using SQL, an application program, or RMU Unload and Load commands to reload the table values. This converted each row to the current version of the table.

This problem has been corrected in V7.0. Oracle Rdb for OpenVMS Alpha now correctly detects null columns for old version rows in a table. ♦

3.1.44 System Table and Index Cardinalities Updated

In previous versions, the cardinalities of some system tables and indexes stored in the system tables, RDB$RELATIONS and RDB$INDICES, were not kept current as the database evolved. This was never a serious problem because of the way Oracle Rdb manages its internal information. However it was occasionally disconcerting to some users.

Oracle Rdb V7.0 now maintains these, as well as other cardinalities, with the same algorithm of approximate cardinality that is used for other tables and indexes.

3.1.45 Database Attach No Longer Leaves Extra Channel Assigned

OpenVMS VAX OpenVMS Alpha In previous versions, during a database attach operation, Oracle Rdb would sometimes leave an extra channel assigned to a disk or network device. Because of this, repeated database detach and attach operations eventually consumed all available channels. This would ultimately cause a database operation to fail with a SYSTEM-F-NOIOCHAN error.

This problem has been corrected in V7.0. Channels are now released correctly. ♦

3.1.46 Bugchecks at PSIISCAN$BWS_SEARCH_SCR + D1 No Longer Occur

In previous versions, the following bugcheck sometimes occurred during backward scans of a sorted index:

```
***** Exception at 00557FC2 : PSIISCAN$BWS_SEARCH_SCR + 000000D1
%COSI-F-BUGCHECK, internal consistency failure
```

The bugcheck was due to an erroneous consistency check. The index was actually consistent.

The following example demonstrates the problem:

```
SQL> CREATE DATABASE FILENAME BWS_PROBLEM;
SQL> CREATE TABLE BWS_TABLE (ID_NUM CHAR(4), DESC_COL CHAR);
SQL> COMMIT;
SQL> BEGIN
cont> DECLARE :ID INT;
cont> WHILE :ID < 1000
cont> LOOP
cont>    INSERT INTO BWS_TABLE VALUES (:ID, ' ');
cont>    SET :ID = :ID + 1;
cont> END LOOP;
cont> END;
SQL> COMMIT;
SQL>
SQL> SELECT COUNT (*) FROM BWS_TABLE;
SQL> CREATE INDEX BWS_INDEX ON BWS_TABLE (ID_NUM, DESC_COL DESC)
cont> TYPE IS SORTED NODE SIZE 86 PERCENT FILL 10;
SQL> COMMIT;
SQL> DELETE FROM BWS_TABLE WHERE id_num > '109' AND id_num < '114';
SQL> COMMIT;
SQL> SELECT * FROM BWS_TABLE WHERE id_num = '110' ORDER BY desc_col;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file ...
%COSI-F-BUGCHECK, internal consistency failure
```

One workaround was to rebuild the index. Another workaround was to restructure the query such that a backward scan was not used on the index.

This problem has been corrected in V7.0.

3.1.47 Invalid Monitor Home Directory No Longer Causes Server Failures

In previous versions, if the directory from which the database monitor was invoked was not valid, any server processes subsequently invoked by that monitor possibly would not be able to create files. For example, a directory is not valid if it no longer exists or the directory specification cannot be parsed because it contains process-local concealed logicals. The files typically created by the database server processes are temporary work files that are created in the home directory, which is inherited from the database monitor.

The problem was identified by a failure of one of the database server processes, usually the AIJ backup server (ABS). It was possible, but less likely, that the other database server processes could be affected by this problem. When the database recovery (DBR) process encountered this problem, the database was shutdown.

The affected server process failed in its attempt to create a file, with the following exception:

```
***** Exception at 0002CB38 : AIJBCK$CREATE_BACKUP_FILE + 000001D4
%RDMS-F-FILACCERR, error parsing name of file SYS$DISK:[].AIJ;
-RMS-F-DIR, error in directory name
```

This error indicated a problem when the ABS process was building the file specification of the .aij file. One step in that process requires parsing the current working directory of the server process. If that directory cannot be successfully parsed, the parse system call fails and the ABS process aborts with a bugcheck.

A server process inherits its current working directory (home directory) from its creator, the database monitor process. The monitor process inherits the home directory from the process that starts it (that is, the user process issuing the RMU Monitor Start or @RMONSTART commands).

The workaround was to make sure the monitor was always started when the current working directory of the starting process was valid, that is, could be successfully parsed. You could achieve this using the following simple DCL:

```
$ IF F$PARSE("SYS$DISK:[]",,,,"SYNTAX_ONLY") .EQS. "" THEN -
      WRITE SYS$OUTPUT "Bad default directory !"
```

Once the problem began, you needed to close all databases, then stop the monitor on the affected node, set default to a valid directory and restart the monitor.

This problem has been corrected in V7.0. Before invoking any server process, the database monitor verifies the validity of its home directory syntax and the existence of the home directory. If there are any problems during the verification process, the following message is written to the monitor log:

```
10-AUG-1995 10:48:53.49 - received AIJ Log Server start request from 22000120:0
  - home directory "CODD$:[ANDERS.WORK.ALS.TEST]" is invalid
  - database monitor created AIJ log server RDM_ALS701_1 (22000134)
  - sending normal AIJ Log Server start reply to requestor
```

The DBR, ALS and record cache server (RCS) processes are invoked regardless of the validity of the monitor's home directory. The LCS, LRS and ABS server processes are *not* invoked if the database monitor's home directory cannot be accessed.

3.1.48 Large Queries No Longer Bugcheck at RDMS$$GEN_ROOM+14

OpenVMS Alpha

In previous versions, when compiling a large query or module, Oracle Rdb sometimes would bugcheck at RDMS$$GEN_ROOM + 00000014.

Note

This error is a generic failure indicator. Fixing one instance of the error does not necessarily mean that all instances have been corrected. If you experience any new occurrences of this bugcheck, please contact your Oracle Supercenter or representative.

The following example shows the bugcheck dump:

```
***** Exception at xxxxxxxx : RDMS$$GEN_ROOM + 00000014
%COSI-F-BUGCHECK, internal consistency failure
```

The only workaround was to break the query into several smaller queries.

This problem has been corrected in V7.0. ♦

3.1.49 Bugcheck at RDMS$$EXE_CREATE_TTBL_FILE+5D Is Fixed

OpenVMS VAX

In previous versions, a bugcheck at RDMS$$EXE_CREATE_TTBL_FILE+5D was returned while evaluating constraints that required a temporary table on OpenVMS VAX. On all platforms, there was a small memory leak that occurred whenever a TTBL was used. Note that the leak was negligible—the query had to run 40000 times before a megabyte would disappear.

This problem has been corrected in V7.0. ♦

3.1.50 File Error Messages During Query Execution Are Now Correct

In V6.1, file error messages were returned during query execution. This was caused by a file system problem, resulting in file error messages that were incomplete or incorrect, and that contained misleading extraneous text.

The following example shows a sample of the file error messages:

```
%RDB-F_IO_ERROR, input or output error
%NONAME-F-NOMSG, Message number 00000004
```

In this example, the problem was that the correct error message, which should have followed the `%RDB-F_IO_ERROR` line, was omitted and the `NOMSG` line was incorrectly added. In all known occurrences of this problem, the first error was `%RDB-F_IO_ERROR` or `%RDB-F-SYS_REQUEST`. However, all cases of `%RDB-F_IO_ERROR` or `%RDB-F-SYS_REQUEST` were not incorrect.

This problem has been corrected in V7.0.

3.1.51 Sort and Merge Routines Are No Longer Called in Incorrect Order

In previous versions, a query may have been terminated by an anticipatable error such as a deadlock. Subsequent execution of the same query may have been terminated with one of the following error messages because sort or merge routines were called in incorrect order:

```
COSI$_SORT_ON, sort or merge routines called in incorrect order
```

```
SOR$_SORT_ON, sort or merge routines called in incorrect order
```

The workaround to this problem was to terminate the program and rerun it or, to detach the database and reattach. This reexecuted the query without error.

This problem has been corrected in V7.0.

3.1.52 RDMS$BIND_WORK_VM Now May Be a Large Value

In previous versions, specifying a virtual memory (VM) value greater than 65,535 bytes for the RDMS$BIND_WORK_VM logical name may have produced incorrect results for some queries. The RDMS$BIND_WORK_VM logical name permits you to reduce the overhead of disk I/O for matching operations.

This problem has been corrected in V7.0. You can now set the VM allocated to your process for use in matching operations to any reasonable large value.

3.1.53 VLDB Application Storage Areas No Longer Exhaust the Channel Limit

In previous versions, it was possible for the storage areas of a very large database (VLDB) to exhaust the previous I/O channel limit of 4096 open channels per database attach. The 4096 open channel limit was actually equivalent to only 2046 storage areas because of the corresponding snapshot areas. This problem involved all interfaces.

The problem occurred most often when you tried to create a database containing more than 2046 storage areas. However, it also occurred at runtime.

For V7.0 on OpenVMS Alpha and Digital UNIX, the maximum number of open I/O channels has been increased from 4096 to 8192.

The new limit allows up to 4094 database storage areas, and their corresponding snapshot areas, to be open simultaneously.

For V7.0 on OpenVMS VAX, the limit of open files has been increased to 2047. This effectively removes the Oracle Rdb limit to the number of open files allowed on OpenVMS VAX systems.

Note that the actual number of files that can be opened by a process on OpenVMS is limited by the OpenVMS SYSGEN parameter CHANNELCNT and the OpenVMS UAF parameter FILLM. Other process quotas and limits may also limit the number of open files for a process.

3.1.54 Data Converted from TEXT to BIGINT (QUADWORD) No Longer Loses Precision with Large Values

The database engine converts TEXT literals to BIGINT by using G_FLOAT as an intermediate format. Prior to V6.1, precision was lost with large values (16-digit precision or more) because the G_FLOAT data type has a limit of 15 decimal digits precision, while a column of data type BIGINT can contain integers with up to 18 decimal digits.

This problem has been corrected in V6.1 and higher.

3.1.55 Changes to TIMESTAMP Literal and Character Format

The SQL interface to Oracle Rdb implemented the SQL-92 date-time support in Version 4.1. However, Version 4.1 of Oracle Rdb actually preceded the official adoption of SQL-92 as the standard. The draft on which V4.1 SQL was based defined the separator between the date and the time portion of the TIMESTAMP literal string as a colon (:). However, the final version of the SQL-92 standard changed this to a space.

In Oracle Rdb V7.0, the input TIMESTAMP format has been enhanced to accept both the SQL-92 standard as well as the format previously accepted.

When a timestamp value is converted to CHAR or VARCHAR, such as when the timestamp value displayed by the interactive SQL SELECT or PRINT statements, SQL now displays it only in the SQL-92 format. The following example shows the use of the old and new TIMESTAMP literal support:

```
SQL> SET DEFAULT DATE FORMAT 'SQL92';
SQL>
SQL> CREATE TABLE T (a TIMESTAMP(2) DEFAULT TIMESTAMP'1995-1-1 12:34:10.01',
cont>                b INTEGER);
SQL> INSERT INTO t (b) VALUE (0);
1 row inserted
SQL> INSERT INTO T (b,a) VALUE (1, TIMESTAMP'1995-1-1:12:34:10.01');
1 row inserted
SQL> INSERT INTO T (b,a) VALUE (2, TIMESTAMP'1995-1-1 12:34:10.01');
1 row inserted
SQL> INSERT INTO T (b,a) VALUE (3, CAST(CAST(TIMESTAMP'1995-1-1:12:34:10.01'
cont>                                  AS VARCHAR(30)) AS TIMESTAMP(2)));
1 row inserted
SQL> INSERT INTO T (b,a) VALUE (4, CAST(CAST(TIMESTAMP'1995-1-1 12:34:10.01'
cont>                                  AS VARCHAR(30)) AS TIMESTAMP(2)));
1 row inserted
SQL> INSERT INTO T (b, a) VALUE (5, CURRENT_TIMESTAMP);
1 row inserted
SQL>
SQL> SELECT b, a, CAST(a AS CHAR(30)) FROM T ORDER BY b;
           B   A
           0   1995-01-01 12:34:10.01   1995-01-01 12:34:10.01
           1   1995-01-01 12:34:10.01   1995-01-01 12:34:10.01
           2   1995-01-01 12:34:10.01   1995-01-01 12:34:10.01
           3   1995-01-01 12:34:10.01   1995-01-01 12:34:10.01
           4   1995-01-01 12:34:10.01   1995-01-01 12:34:10.01
           5   1995-06-26 16:12:08.95   1995-06-26 16:12:08.95
6 rows selected
SQL>
SQL> CREATE DOMAIN TEXT_BUFF CHAR(30);
SQL>
SQL> BEGIN
cont>     DECLARE :X TEXT_BUFF;
cont>     SET :X = CAST(CURRENT_TIMESTAMP AS TEXT_BUFF);
cont>     TRACE :X;
cont> END;
~Xt: 1995-06-26 16:12:09.11
SQL>
```

This enhancement should have little or no impact on your applications. Oracle Rdb continues indefinitely to support the non-standard TIMESTAMP format. This change may affect applications that parse the formatted TIMESTAMP values, if these applications (or routines) expect a separating colon. The applications should be modified to accept either a space or a colon. This change may also affect the RMU Unload and RMU Load commands that use delimited flat files with a space character as a delimiter. If possible, use quoting around the data values in the delimited format to avoid ambiguity.

3.1.56 External Functions Now Produce Valid Descriptor Lengths

OpenVMS VAX OpenVMS Alpha

In previous versions, use of an external function to process CHAR parameters by descriptor sometimes produced invalid descriptor lengths in the external function routine.

When the length of the source expression exceeded the size of the declared external function parameter, Oracle Rdb incorrectly passed the actual expression length instead of limiting the length to that of the format routine parameter. Only when the source is shorter than the function parameter should Oracle Rdb modify the descriptor length.

The workaround to this problem was to use the built-in CAST function, which converts a value expression to another type of data, to restrict the size of the input expressions to the size of the external function parameter. The following is an example of the workaround:

```
SQL> -- Use a domain for consistent usage.
SQL> CREATE DOMAIN text_buffer CHAR(10);
SQL> -- Create a function using the domain.
SQL> CREATE FUNCTION mytest (text_buffer BY DESCRIPTOR)
cont>    RETURNS INTEGER BY VALUE;
cont>    EXTERNAL LOCATION 'mytest.exe'
cont>             LANGUAGE GENERAL
cont>             GENERAL PARAMETER STYLE;
cont> -- When calling the function, use CAST to limit the size.  Use the
cont> -- domain to guarantee the same size data as required for the function.
SQL> SELECT mytest(CAST(JOB_TITLE AS text_buffer)) FROM JOBS LIMIT TO 1 ROW;
```

This problem has been corrected in V7.0. The descriptor length is limited to the range 0 to the size of the function CHAR parameter. The following example shows the function MYTEST, which simply prints the length as passed in the string descriptor:

```
SQL> ATTACH 'FILENAME personnel';
SQL> CREATE FUNCTION mytest (CHAR (10) BY DESCRIPTOR
cont>   RETURNS INTEGER BY VALUE;
cont>   EXTERNAL LOCATION 'mytest.exe'
cont>   LANGUAGE GENERAL
cont>   GENERAL PARAMETER STYLE;
SQL>
SQL> SHOW ALL DOMAINS job_title;
JOB_TITLE                       CHAR(20)
 Comment:        Generic job title
 Missing Value: None
SQL> SELECT job_title FROM JOBS
cont>   WHERE job_title STARTING WITH 'Dept' OR
cont>         job_title STARTING WITH 'Assoc';
 JOB_TITLE
 Associate Programmer
 Dept. Supervisor
2 rows selected
SQL>
SQL> SELECT mytest(job_title)  FROM JOBS
cont>   WHERE job_title STARTING WITH 'Dept' OR
cont>         job_title STARTING WITH 'Assoc';

          10
          10
2 rows selected
SQL>
SQL> SELECT mytest(CAST(job_title AS CHAR(40000)))
cont>   FROM JOBS LIMIT TO 1 ROW;

          10
1 row selected
SQL>
SQL> SELECT mytest(CAST(job_title AS CHAR(10)))
cont>   FROM JOBS LIMIT TO 1 ROW;

          10
1 row selected
SQL>
SQL> SELECT mytest(CAST(job_title AS CHAR(5)))
cont>   FROM JOBS LIMIT TO 1 ROW;

           5
1 row selected
```

```
SQL>
SQL> SELECT mytest(CAST(job_title AS CHAR(0)))
cont>   FROM JOBS LIMIT TO 1 ROW;

          0
1 row selected
```

♦

3.1.57 SPAM Page Search Algorithm Is Now Optimized

In previous versions, the space area management (SPAM) page search algorithm was not optimized to prevent excessive searching of the storage area.

This problem occurred more frequently under the following circumstances:

- The number of users attached to the database was large.
- The number of buffers allocated to each user was large.
- The SPAM page interval was large.
- Mixed page format storage areas for hashed indexes were used, particularly when the hashed indexes were close to the fullness threshold.

However, even in databases that did not have these attributes, excessive searching of the storage area did occur.

Although there was no certain workaround, there were a number of adjustments that could have minimized the problem, including the following:

- Increase the allocation size of the affected storage area.
- Decrease the SPAM page interval.
- Reduce the number of buffers per user.

This problem has been corrected in V7.0. The SPAM page search algorithm has been optimized to prevent excessive searching of the storage area.

3.1.58 RDMS$BIND_SEGMENTED_STRING_COUNT or RDB_BIND_SEGMENTED_STRING_COUNT No Longer Causes VM Corruption

In previous versions, using the logical name RDMS$BIND_SEGMENTED_STRING_COUNT or the configuration parameter RDB_BIND_SEGMENTED_STRING_COUNT sometimes led to virtual memory corruption which could have led to unexpected bugcheck dumps from Oracle Rdb.

Oracle Rdb uses this logical name or configuration parameter to preallocate data structures when processing tables with many segmented strings. Oracle Rdb was allocating only enough space for 80% of the count. The algorithm was using the wrong structure size. As a result, after the allocated 80% was used, all future references wrote on memory allocated for other uses.

The workaround was to increase the value of the logical name or configuration parameter so that the top 20% of the buffer was not used, or to avoid using this logical name.

This problem is corrected in V7.0.

3.1.59 DEC MMS and CDD/Repository Report EXEDELPROC

OpenVMS VAX OpenVMS Alpha

DEC MMS, when used with Oracle CDD/Repository, gave the error "EXEDELPROC, Subprocess terminated abnormally." This is a problem in DEC MMS, which a change to Oracle Rdb recently uncovered. A change in Oracle Rdb V6.0A ECO 4 and V6.1 ECO 3 made the problem visible.

Oracle Rdb has been modified in V7.0 so that it does not expose the bug in DEC MMS. ♦

3.2 SQL Errors Fixed

This section describes problems that have been fixed in the SQL interface.

3.2.1 LIBSQL Naming Conflict Corrected

Digital UNIX

The Oracle Rdb installation creates a symbolic link from the /usr/lib/libsql.so file to the shared object libsql.so, which is used in link commands for SQL module processor and SQL precompiler applications. A conflict can occur when more than one SQL product is installed on the same UNIX machine. If two products, such as Oracle Rdb and Oracle7, have been installed on the same Digital UNIX machine and both products contain a libsql.so file, the most recent installation resets the link to its libsql.so file.

This problem has been corrected in V7.0. To resolve this problem, Oracle Rdb changed the link to librdbsql.so. That is, /usr/shlib/librdbsql.so is a link to the libsql.so file in the Oracle Rdb path.

If you have applications from previous versions that include –lsql in the link command, you should change –lsql to –lrdbsql for Oracle Rdb SQL applications. ♦

3.2.2 Full Outer Join with Derived Tables and IS NULL Predicate No Longer Returns Incorrect Results

In previous versions, a full outer join with derived tables and the IS NULL predicate sometimes returned incorrect results, as shown in the following example:

```
-- Create sample database and load some records.
--
CREATE DATA FILE FOO;
CREATE TABLE A
  (A1 CHAR(10),
   A2 CHAR(10));
CREATE TABLE B
  (B1 CHAR(10),
   B2 CHAR(10));
INSERT INTO A (A1,A2) VALUES ('row1','a1 row1');
INSERT INTO A (A1,A2) VALUES ('row2','a1 row2');
INSERT INTO A (A2) VALUES ('a1 null');
INSERT INTO B (B1,B2) VALUES ('row2','b1 row2');
INSERT INTO B (B1,B2) VALUES ('row1','b1 row1');
INSERT INTO B (B2) VALUES ('b1 null');
COMMIT;
```

```
--
-- Display contents of table A.
--
SELECT * FROM A;
 A1           A2
 row1         a1 row1
 row2         a1 row2
 NULL         a1 null
3 rows selected
--
-- Display contents of table B.
--
SELECT * FROM B;
 B1           B2
 row2         b1 row2
 row1         b1 row1
 NULL         b1 null
3 rows selected
--
-- Display contents of a full outer join with derived tables.
--
SELECT A.A1, A.A2, B.B1, B.B2 FROM
     (SELECT * FROM A) A
      FULL OUTER JOIN
      (SELECT * FROM B) B
     ON B.B1 = A.A1;
 A.A1         A.A2         B.B1         B.B2
 row1         a1 row1      row1         b1 row1
 row2         a1 row2      row2         b1 row2
 NULL         NULL         NULL         b1 null
 NULL         a1 null      NULL         NULL
4 rows selected
--
-- Display the result of the query where A1 is null.
-- ** Do not get expected result
-- and a conjunct is at the wrong place. **
--
SELECT A.A1, A.A2, B.B1, B.B2 FROM
     (SELECT * FROM A) A
      FULL OUTER JOIN
      (SELECT * FROM B) B
     ON B.B1 = A.A1
   WHERE A1 IS NULL;
Match    (Full Outer Join)
  Outer loop
    Sort
    Merge of 1 entries
      Merge block entry 1
      Conjunct        Get     Retrieval sequentially of relation A
                       -----THIS CONJUNCT IS NOT CORRECT
  Inner loop
    Temporary relation      Sort
    Merge of 1 entries
      Merge block entry 1
      Get     Retrieval sequentially of relation B
 A.A1         A.A2         B.B1         B.B2
 NULL         NULL         row1         b1 row1
 NULL         NULL         row2         b1 row2
 NULL         NULL         NULL         b1 null
 NULL         a1 null      NULL         NULL
4 rows selected
```

The IS NULL predicate in the WHERE clause was pushed down below the full outer join and caused the query to return wrong results (see the conjunct highlighted in the preceding example). This predicate should have been applied on top of the full outer join.

This problem has been corrected in V7.0.

3.2.3 Assignment Statement No Longer Uses Incorrect Value for CURRENT_TIMESTAMP

In previous versions, Oracle Rdb, in some cases, incorrectly processed assignment (SET) statements in multistatement and stored procedures that referred to the date/time functions CURRENT_TIME, CURRENT_DATE, and CURRENT_ TIMESTAMP.

Oracle Rdb used the time, date or timestamp of the procedure start instead of the date and time of the current statement. This problem was only seen when SET statements occurred at the end of a compound statement.

Other statements, such as TRACE, INSERT, SELECT, UPDATE, and subqueries in conditional statements were not affected by this problem.

This problem has been corrected in V7.0.

3.2.4 COMPUTED BY Column Value Now Returned Correctly

In previous versions, references to the value of a COMPUTED BY column that contained a subquery (SELECT expression) sometimes returned the wrong result.

This problem occurred in the following situations:

- In a FOR loop or singleton SELECT within an outer FOR loop.

  The reference to the COMPUTED BY column from a table in the outer FOR loop returned the wrong result.

  The following example shows the problem for two nested FOR loops in a multistatement procedure. The inner FOR loop fetches zero rows on the first iteration of the outer loop, when it should fetch a single matching row. The value fetched on the second iteration is the value expected during the first iteration.

```
SQL> set flags 'TRACE';
SQL>
SQL> create table FCOL_1 (a int, b int);
SQL> insert into FCOL_1 (a, b) values (1, 100);
1 row inserted
SQL> insert into FCOL_1 (a, b) values (2, 200);
1 row inserted
SQL>
SQL> create table FCOL_2 (a int, c computed by (select f.b
cont>                                            from FCOL_1 f
cont>                                            where FCOL_2.a = f.a));
SQL> insert into FCOL_2 (a) value (1);
1 row inserted
SQL> insert into FCOL_2 (a) value (2);
1 row inserted
SQL>
SQL> create table FCOL_3 (a int, d int);
SQL> insert into FCOL_3 (a, d) value (100, -100);
1 row inserted
SQL> insert into FCOL_3 (a, d) value (200, -200);
1 row inserted
SQL>
SQL> begin
cont> for :a as
cont>     select a, c
cont>     from FCOL_2
cont> do
cont>     begin
cont>     declare :x int = :a.c;
cont>     trace '------ :a.A ', :a.a, ' :a.C ', :a.c;
cont>     for :b as
cont>         select d
cont>         from FCOL_3
cont>         where a = :a.c
cont>     do
cont>         trace ':b.D ', :b.d;
cont>     end for;
cont>     end;
cont> end for;
cont> end;
~Xt: ------ :a.A 1           :a.C 100
~Xt: ------ :a.A 2           :a.C 200
~Xt: :b.D -100
```

The expected result was "~Xt: :b.D -100" for the first iteration, and "~Xt: :b.D -200" for the second iteration of the loop.

- Using an AFTER INSERT or AFTER UPDATE trigger that referred to a COMPUTED BY column in the triggering table.

  The following example shows that the value inserted by the AFTER INSERT trigger is incorrect:

```
SQL> CREATE TABLE INTERMEDIATE (FLD1 CHAR(10));
SQL> CREATE TABLE TRIGGER_RESULT (FLD1 CHAR(10));
SQL> CREATE TABLE TEST_COMPUTED
cont>     (FLD1 CHAR(10),
cont>      FLD2 COMPUTED BY (SELECT FLD1
cont>            FROM INTERMEDIATE
cont>            LIMIT TO 1 ROW));
SQL> INSERT INTO INTERMEDIATE VALUES ('string1');
1 row inserted
SQL> SELECT * FROM TEST_COMPUTED;
0 rows selected
SQL> INSERT INTO TEST_COMPUTED (FLD1) VALUES ('string2');
1 row inserted
```

```
SQL> SELECT * FROM TEST_COMPUTED;
 FLD1         FLD2
 string2      string1
1 row selected
SQL> CREATE TRIGGER TEST_COMPUTED_TRIGGER
cont>     AFTER INSERT ON TEST_COMPUTED
cont>     (INSERT INTO TRIGGER_RESULT (FLD1)
cont>      VALUES (TEST_COMPUTED.FLD2))
cont>     FOR EACH ROW;
SQL> INSERT INTO TEST_COMPUTED (FLD1) VALUES ('string3');
1 row inserted
SQL> SELECT * FROM TRIGGER_RESULT;
 FLD1
 ..........
1 row selected
SQL> rollback;
```

The expected value in the table TRIGGER_RESULT was "string1". The value displayed by interactive SQL indicates that the string contains unprintable characters.

This problem has been corrected in V7.0.

3.2.5 Computed By Column Now Set to Null During DROP TABLE CASCADE

In previous versions, a DROP TABLE CASCADE statement failed if the table being dropped was referred to in a COMPUTED BY column in another table.

In V7.0, Oracle Rdb sets the COMPUTED BY column to NULL if it refers to a table that has been deleted by a DROP TABLE CASCADE statement. For example:

```
SQL> CREATE TABLE t1 (col1 INTEGER,
cont>                 col2 INTEGER);
SQL> --
SQL> CREATE TABLE t2 (x INTEGER,
cont>                 y COMPUTED BY (SELECT COUNT(*) FROM
cont>                   t1 WHERE t1.col1 = t2.x));
SQL> --
SQL> -- Assume values have been inserted into the tables.
SQL> --
SQL> SELECT * FROM t1;
        COL1         COL2
           1          100
           1          101
           1          102
           2          200
           3          300
5 rows selected
SQL> SELECT * FROM t2;
           X            Y
           1            3
           3            1
2 rows selected
SQL> --
SQL> DROP TABLE t1 CASCADE;
Computed Column Y in table T2 is being set to NULL.
SQL> SELECT * FROM t2;
           X            Y
           1            NULL
           3            NULL
```

You can alter the table and drop the COMPUTED BY column. You can later alter the table to create a new COMPUTED BY column of the same name which has a different computed expression.

3.2.6 Unexpected RDMS-F-BAD_SYM Error When Referring to COMPUTED BY Columns Fixed

In previous versions, you could have received an RDMS-F-BAD_SYM error when you accessed a table that referenced views through COMPUTED BY columns. You could have avoided the error by changing the query to reverse the order of the columns selected, as shown in the following example:

```
SQL> CREATE TABLE t (A INTEGER);
SQL> INSERT INTO t VALUES (1);
1 row inserted
SQL> INSERT INTO t VALUES (2);
1 row inserted
SQL> CREATE VIEW v AS SELECT A FROM T;
SQL> CREATE TABLE s (A INTEGER,
cont>           B COMPUTED BY (SELECT MIN(A) FROM v),
cont>           C COMPUTED BY (SELECT AVG(A) FROM v)
cont>           );
SQL> INSERT INTO s (A) VALUES (1);
1 row inserted
SQL>
SQL> SELECT B, C FROM s;
%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-BAD_SYM, unknown field symbol - C
SQL> SELECT C, B FROM s;
                       C             B
  1.500000000000000E+000             1
1 row selected
SQL> rollback;
```

This problem was caused by Oracle Rdb trying to find the referenced column in the view, instead of the table referenced in the query. If the view contained a column of the same name, incorrect results could also be observed.

For this problem to occur, the following unusual characteristics must have been used together:

- The table contained at least two COMPUTED BY columns.
- At least one of these computed by columns was a subquery.
- The subquery referenced a view table.
- The COMPUTED BY column that referenced the view through a subquery was fetched before any other computed by column in the select list.

The only workaround for this problem was to change the column selection order. That is, select the first COMPUTED BY column which meets the characteristics listed as the last in the selection list.

This problem has been corrected in Oracle Rdb V7.0.

3.2.7 FETCH No Longer Returns End-Of-Stream Condition on NO_RECORD

In previous versions, a NO_RECORD condition returned from Oracle Rdb during a FETCH operation was translated to an End-Of-Stream condition. This was incorrect. This problem has been corrected in V7.0. The SQL behavior has been changed so that an UDCURDEL exception is now raised.

3.2.8 OUT Parameters Now Accessible with TRACE Statement

In previous versions, you could not use the TRACE statement to trace the contents of an OUT parameter. Only parameters with modes IN or INOUT could appear in a TRACE statement. Attempts to use an OUT parameter directly or within an expression resulted in the error which is shown in the following example:

```
SQL> CREATE MODULE M1
cont>     LANGUAGE SQL
cont>     PROCEDURE P1 (IN :A INTEGER, OUT :B REAL);
cont>     BEGIN
cont>       SET :B = :A;
cont>       TRACE :A, :B;
cont>     END;
cont> END MODULE;
%SQL-F-NOTINPARAM, Parameter B is referenced as an IN parameter, but
declared as OUT
```

This restriction has been lifted in V7.0. You can use an OUT parameter in a TRACE statement, as shown in the following example. Please note that in all other statements, an OUT parameter must be the target for an assignment, never a source.

```
SQL> CREATE MODULE M1
cont>     LANGUAGE SQL
cont>     PROCEDURE P1 (IN :A INTEGER, OUT :B REAL);
cont>     BEGIN
cont>       SET :B = :A;
cont>       TRACE :A, :B;
cont>     END;
cont> END MODULE;
SQL>
SQL> SET FLAGS 'TRACE';
SQL> DECLARE :RES REAL;
SQL> CALL P1 (10, :RES);
~Xt: 10         1.0000000E+01
          RES
  1.0000000E+01
SQL>
```

3.2.9 Enhanced Support for Views by ALTER DOMAIN and ALTER TABLE Statements

In previous versions, if you modified a domain or a column of a table, a dependent view was left as it was originally defined. That is, its dependent columns retained the data type of the column as it was defined when the view was initially created.

As a result, changes to the table (either directly with ALTER TABLE . . . ALTER COLUMN, or indirectly with ALTER DOMAIN) were not reflected in the view definition. You had to drop the view definition and re-create it to inherit these changes.

The following changes in Oracle Rdb correct this behavior:

- ALTER DOMAIN

  When you modify a domain, its attributes are automatically propagated to all referencing tables and views. In previous versions, only referencing tables inherited all the changed attributes.

For example, if you modify the data type of a domain, Oracle Rdb updates any view column that refers to that domain, directly or indirectly, to reflect the new attributes of that domain. (A view column can refer indirectly to a domain by using an expression that refers to a base table's column which uses that domain.)

- ALTER TABLE . . . ALTER COLUMN

  When you modify the data type or domain of a column, Oracle Rdb automatically propagated those attributes to all referencing views. In previous versions, views did not inherit the changed attributes.

  For example, if you modify a column to refer to a domain for its data type, any view column which refers to that column directly is updated to reflect the new attributes of the modified column.

```
SQL> CREATE DOMAIN d CHAR(1);
SQL> CREATE TABLE T (A d);
SQL> CREATE VIEW V (B,C) AS SELECT A, A||'X' FROM T;
SQL> SHOW VIEW (COL) V;
Information for table V

Columns for view V:
Column Name                     Data Type        Domain
-----------                     ---------        ------
B                               CHAR(1)
C                               CHAR(2)
SQL> ALTER DOMAIN d CHAR(5);
SQL> SHOW VIEW (COL) v;
Information for table V

Columns for view V:
Column Name                     Data Type        Domain
-----------                     ---------        ------
B                               CHAR(5)
C                               CHAR(6)
```

In previous versions, the character length remained as CHAR(2) for column C in view V. In V7.0, column C correctly reflects a data type of CHAR(6). The previous workaround to this problem was to drop the dependent view and re-create it.

3.2.10 Reserved Tables No Longer Removed from Reserving List After ALTER INDEX

In previous versions, when you specified a table in the RESERVING clause of the SET TRANSACTION statement, statements sometimes could not access the table after an index was altered. For example:

```
SQL> SET TRANSACTION READ WRITE RESERVING TEST FOR EXCLUSIVE WRITE;
SQL> ALTER INDEX TEST_HSH STORE IN TEST2;
SQL> ALTER STORAGE MAP TEST_MAP
cont>  STORE IN TEST2;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-UNRES_REL, relation TEST in specified request is not a relation reserved
in specified transaction
```

This error occurred because the table reloaded code for the ALTER INDEX statement (the statement prior to the error). After the index was altered, Oracle Rdb loaded a new version of the tables, but it did not propagate the indication that this table was specified in a RESERVING clause. This has been corrected in V7.0.

3.2.11 CREATE TRIGGER or CREATE MODULE Statement No Longer Generates Unexpected SEGTOOBIG Error

In previous versions, some CREATE statements (usually CREATE TRIGGER or CREATE MODULE) failed with the following error:

```
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-F-IMP_EXC, facility-specific limit exceeded
-RDMS-E-SEGTOOBIG, segmented string segment exceeds maximum allowed size
```

When a trigger, module, view, or constraint object is created, Oracle Rdb stores the original SQL source along with an internal representation (known as BLR). The BLR representation is usually much more compact than the original source.

The failure described here usually occurred because the SQL source exceeded an internal limit imposed by previous versions. Namely, the maximum source length was restricted to 65535 bytes (the largest value for an unsigned word (16 bits)). This limitation has been removed in Oracle Rdb V7.0. The lengths of sources and comments are now limited to 4294967295 bytes (an unsigned longword (32 bits)).

In V6.0 and V6.1, the limit was usually applied by truncating the SQL source string to 65535 bytes before storage. Unfortunately, there is also some record storage system overhead that needs to be counted. Therefore, the correct truncation should have been something less than 65535. The actual length depends on the current setting of the RDMS$USE_OLD_SEGMENTED_STRING logical name, which changes the layout used by the record storage system.

Note that truncating the SQL source did not affect the correct behavior of the trigger or other object. You could use the RMU Extract command to extract large triggers, translating the BLR representation into SQL.

The workaround for this problem was often as simple as trimming trailing spaces and replacing leading spaces with TAB characters (for the same formatting affect) or reducing the number of lines of SQL source used to define the object.

V7.0 removes this restriction.

3.2.12 GET DIAGNOSTICS Statement Now Processed Correctly

In previous versions, Oracle Rdb processed the GET DIAGNOSTICS statement incorrectly if it appeared with other assignment statements in a compound statement. The results of the GET DIAGNOSTICS statement were made available *after* the SET statements. If the SET statements expected to use the result of the GET DIAGNOSTICS statement, incorrect results were observed. This error only occurred when you used SET DIALECT 'SQL92'.

The following example shows that the TOT variable was not correctly updated during execution of the compound statement:

```
SQL> SET DIALECT 'sql92';
SQL> ATTACH 'filename your';
SQL> DECLARE :TOT INT;
SQL> DECLARE :RC INT;
SQL> BEGIN
cont> SET :TOT =  100;
cont> SET :RC = 0;
cont> FOR :C AS EACH ROW OF SELECT k,d FROM t
cont> DO
cont>     DELETE FROM t WHERE k = :c.k;
cont>     GET DIAGNOSTICS :RC = row_count;
cont>     SET :TOT =  :rc;
cont> END FOR;
cont> END;
SQL> PRINT :TOT, :RC;
         TOT           RC
           0            1
```

This problem was the result of an optimization for SELECT statement column assignments that was incorrectly applied to SET statements in a compound statement. The optimization always placed the GET DIAGNOSTICS code at the end of the assignment list, which is important when detecting NULL elimination during aggregate processing of a SELECT statement.

A workaround was to follow the GET DIAGNOSTICS statement with another statement (other than a SET or a GET DIAGNOSTICS statement). This caused the optimization to be disabled and the correct results were reported.

This problem has been corrected in V7.0.

3.2.13 RETURNED_SQLSTATE and RETURNED_SQLCODE No Longer Incorrect After COMMIT, ROLLBACK, and SET TRANSACTION Statements

In previous versions, the SET TRANSACTION, COMMIT, and ROLLBACK statements within a compound statement did not reset SQLCODE during execution. Therefore, a GET DIAGNOSTICS statement that retrieved the RETURNED_SQLCODE or RETURNED_SQLSTATE after these statements sometimes reported the result of a previous INSERT, UPDATE, SELECT, or DELETE statement.

This problem has been corrected in V7.0. Now, SQLSTATE and SQLCODE are returned with success values regardless of the status of the statement before the COMMIT, ROLLBACK, or SET TRANSACTION.

3.2.14 Queries with Expressions Containing Variables No Longer Return Wrong Results

In previous versions, because of early evaluation of queries that contained expressions that in turn contained variables, Oracle Rdb returned a wrong result the first time the query ran. The result was correct for subsequent iterations. The following shows an example that generated a wrong result:

```
SELECT COUNT(*) INTO :y  FROM t1,t2
       WHERE t1.f1 = (:x * 10)
             AND t1.f1= t2.f1;
```

Using query outlines to change the strategy was a workaround to the problem.

This problem has been corrected in V7.0 by using lazy expression evaluation instead of early expression evaluation. As a result, the expression containing the host variable is evaluated at the time when its computed value is needed to evaluate the selection predicate.

3.2.15 Invalid DATE and TIMESTAMP Literals No Longer Accepted

In previous versions, Oracle Rdb did not check the DATE or TIMESTAMP literal format correctly in some cases and allowed invalid and misleading dates or timestamps to be stored in the database. The following shows some examples:

```
SQL> SELECT DATE '0000-00-00' FROM RDB$DATABASE;

 1858-11-17
1 row selected
SQL>
SQL> SELECT DATE '0000-00-01' FROM RDB$DATABASE;

 0000-00-01
1 row selected
SQL>
SQL> SELECT TIMESTAMP '0000-00-00 00:00:00.01' FROM RDB$DATABASE;

 0000-00-00 00:00:00.01
1 row selected
```

This problem has been corrected in V7.0. Literals of type DATE or TIMESTAMP cannot have the year, or month or day set to zero.

3.2.16 Searched Update and Searched Delete Statements No Longer Promote Locks Excessively

In V6.0 and V6.1, searched UPDATE and searched DELETE statements in SQL performed excessive lock promotions on rows fetched for update or delete. This was not the case with previous versions such as V4.2 or V5.1.

Oracle Rdb V6.0 reverted to fetching rows for SHARED READ and then promoting the locks to EXCLUSIVE WRITE. This change in behavior could have resulted in reported deadlocks when none were expected, and certainly led to higher CPU usage than expected.

Workarounds to this problem included using the RESERVING clause to lock the table at a higher mode, or using an UPDATE ONLY table cursor to fetch the rows and using the UPDATE . . . WHERE CURRENT OF or DELETE . . . WHERE CURRENT OF syntax to perform the update or delete operation.

This problem has been corrected in Oracle Rdb V7.0. The searched UPDATE and searched DELETE statements now fetch rows with an EXCLUSIVE WRITE lock. The result is fewer lock promotions, and thus better deadlock avoidance.

3.2.17 Single Area WITH LIMIT Storage Map Now Used Correctly

In previous versions, simple storage maps with a single storage area and STORE USING . . . WITH LIMIT syntax were not handled correctly. Oracle Rdb erroneously assumed that a single area meant that the map had no WITH LIMIT clause.

The insert operation in the following example succeeded even though the value is outside the map limit:

```
CREATE TABLE TEST_LIMIT(CHAR1 CHAR(5), CHAR2 CHAR(5));

CREATE STORAGE MAP LIMIT_MAP
    FOR TEST_LIMIT STORE USING (CHAR1)
 IN "TEST" WITH LIMIT OF ('m');

INSERT INTO TEST_LIMIT(CHAR1) VALUES('n') RETURNING DBKEY;
```

This problem has been corrected in V7.0. When you omit the OTHERWISE clause from a storage map definition, Oracle Rdb uses the WITH LIMIT clause to restrict the values inserted into the table, even for storage maps with one storage area.

3.2.18 ALTER STORAGE MAP Now Lets You Remove the USING Column

In previous versions, if you created a storage map to partition on a particular column in the table, but later wanted to remove the column from the table, Oracle Rdb returned errors.

The following example shows this previous, incorrect behavior:

```
SQL> CREATE STORAGE MAP TEST_STORAGE_MAP for TEST_TABLE
cont>    USING (TEST_COLUMN)
cont>    WITH LIMIT OF (100) in TEST_STORAGE_AREA_A
cont>    OTHERWISE IN TEST_STORAGE_AREA_B;
SQL>
SQL> ALTER STORAGE MAP TEST_STORAGE_MAP
cont>    STORE IN TEST_STORAGE_AREA_A
cont>    REORGANIZE AREAS;
SQL>
SQL> ALTER TABLE TEST_TABLE DROP TEST_COLUMN;
-RDMS-F-FLDINSTO, field TEST_COLUMN is referenced in storage map
TEST_STORAGE_MAP
-RDMS-F-RELFLDNOD, field TEST_COLUMN has not been deleted from relation
TEST_TABLE
```

This problem has been corrected in Oracle Rdb V7.0.

3.2.19 Unexpected UNRES_REL Error No Longer Occurs When Reserving Views

In previous versions, Oracle Rdb did not correctly reserve some complex views when they were used in the RESERVING clause of DECLARE or SET TRANSACTION statements.

When a view is reserved, Oracle Rdb propagates the RESERVING clause to all base tables to which the view refers. If a view in a column subquery expression referred to a table, the table was reserved correctly only during the first transaction.

The following query shows the problem. The base table, T_CODES, is referenced by a view used in a subquery within the reserved view, V_TESTVIEW_3.

```
SQL> SET TRANSACTION READ ONLY RESERVING V_TESTVIEW_3 FOR SHARED READ;
SQL> SELECT * FROM V_TESTVIEW_3;
 VIEW_VALUE   VIEW_DESCRIPTION
 A            character A
1 row selected
SQL> COMMIT;
SQL>
SQL> SET TRANSACTION READ ONLY RESERVING V_TESTVIEW_3 FOR SHARED READ;
SQL> SELECT * FROM V_TESTVIEW_3;
%RDB-E-UNRES_REL, relation T_CODES in specified request is not a relation
reserved in specified transaction
SQL> COMMIT;
```

This problem has been corrected V7.0. Oracle Rdb now correctly reserves base tables to which reserved views refer.

3.2.20 Initialize Handles and External Globals Command Line Qualifiers Processed Correctly

The command line qualifiers that control initializing alias handles were not processed correctly in V6.1. You specify these options with precompiled SQL and SQL module language as follows:

OpenVMS VAX OpenVMS Alpha

On OpenVMS:

```
$ SQLPRE /SQLOPT=(INITIALIZE_HANDLES,EXTERNAL_GLOBALS)
$ SQLMOD /INITIALIZE_HANDLES /EXTERNAL_GLOBALS
```
♦

Digital UNIX

On Digital UNIX:

```
% sqlpre -s '-init -extern'
% sqlmod -init -extern
```
♦

These options can be negated (disabled) by preceding them with "no".

When the INITIALIZE_HANDLES or –init option is enabled, SQL should initialize the alias handle for each alias definition (an alias declared with a DECLARE ALIAS statement that does not specify the EXTERNAL keyword). When disabled, SQL should not initialize alias handles. The INITIALIZE_HANDLES and –init option was supposed to be enabled by default and was to take precedence over the EXTERNAL_GLOBALS or –extern option if any combination of the two were simultaneously specified.

When the EXTERNAL_GLOBALS or –extern option is enabled, SQL should treat an alias reference (an alias declared with a DECLARE ALIAS statement that specifies the EXTERNAL keyword) as an alias definition. As such, the alias handle was supposed to be initialized in the same way as the handle of an alias definition. The EXTERNAL_GLOBALS option was supposed to be enabled by default on OpenVMS platforms for compatibility with previous versions, and disabled by default on Digital UNIX platforms.

V6.1 erroneously processed the EXTERNAL_GLOBALS or –extern option: only aliases declared *without* the LOCAL, EXTERNAL, or GLOBAL keywords were affected. The alias handles of such aliases were not initialized, while the alias handles for LOCAL, EXTERNAL and GLOBAL aliases were initialized when the EXTERNAL_GLOBALS or –extern option was enabled. When the EXTERNAL_GLOBALS or –extern option was disabled, all alias handles were initialized. Moreover, V6.1 erroneously treated the negation of the INITIALIZE_HANDLES or –init option as though the EXTERNAL_GLOBALS or –extern option was enabled.

The following series of examples shows the described problems.

Assume that the following statements are included in an SQL precompiled program EXAMPLE.SC:

```
DECLARE DEFAULT_ALIAS          ALIAS FOR FILENAME 'PERSONNEL';
DECLARE LOCAL_ALIAS    LOCAL    ALIAS FOR FILENAME 'PERSONNEL';
DECLARE EXTERNAL_ALIAS EXTERNAL ALIAS FOR FILENAME 'PERSONNEL';
DECLARE GLOBAL_ALIAS   GLOBAL   ALIAS FOR FILENAME 'PERSONNEL';
```

OpenVMS VAX OpenVMS Alpha In previous versions, on OpenVMS, if you used the following command, the alias handles for LOCAL_ALIAS, EXTERNAL_ALIAS and GLOBAL_ALIAS were initialized, but they should not have been:

```
$ SQLPRE/CC EXAMPLE.SC /SQLOPT=NOINITIALIZE_HANDLES
```

If you used the following command, the alias handle for EXTERNAL_ALIAS was initialized, but should not have been:

```
$ SQLPRE/CC EXAMPLE.SC /SQLOPT=NOEXTERNAL_GLOBALS
```

If you used the following command, the alias handle for DEFAULT_ALIAS was not initialized, but should have been:

```
$ SQLPRE/CC EXAMPLE.SC /SQLOPT=EXTERNAL_GLOBALS
```
♦

Digital UNIX On Digital UNIX, if you used the following command, the alias handles for LOCAL_ALIAS, EXTERNAL_ALIAS and GLOBAL_ALIAS were initialized, but they should not have been:

```
sqlpre -l cc EXAMPLE.SC -s '-noinit'
```

If you used the following command, the alias handle for EXTERNAL_ALIAS was initialized, but should not have been:

```
sqlpre -l cc EXAMPLE.SC -s '-noextern'
```

If you used the following command, the alias handle for DEFAULT_ALIAS was not initialized, but should have been:

```
sqlpre -l cc EXAMPLE.SC -s '-extern'
```
♦

These problems have been corrected in Oracle Rdb 7.0.

3.2.21 SQL Precompiler Now Consistent for C and COBOL Symbolic Debugging

Digital UNIX For applications running on Digital UNIX, the SQL precompiler previously compiled C and COBOL source language files inconsistently regarding symbolic debugging.

By default, C language source files were compiled for full symbolic debugging, including suppression of any optimizations that would limit full symbolic debugging; COBOL language source files were compiled without producing symbol table information for symbolic debugging. This inconsistency is corrected in V7.0. By default, both C and COBOL source files are compiled without producing symbol table information for symbolic debugging.

3.2.22 Bugcheck Creating Complex Views Fixed

OpenVMS Alpha In previous versions, when creating a complex view, two bugchecks could have occurred: an RDSBUGCHK and an SQLBUGCHK. This situation only occurred on the OpenVMS Alpha platform and occurred in one of the routines for allocating memory. The following example shows exceptions that could have occurred:

```
---- RDSBUGCHK.DMP;1

***** Exception at 00CCF84C : RDMS$$COMPILE_STMT + 0000A68C
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=00000144,
PC=00CCF84C, PS=0000000B

---- SQLBUGCHK.DMP;1
```

```
***** Exception at 000D1854 : AMAC_FLT_CVTLG + 000002D4
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=00000144,
PC=000D1854, PS=0000001B
```

If a VAX system was available, the view could have been created successfully from that platform.

This problem has been corrected in Oracle Rdb V7.0. ♦

3.2.23 SQL Now Generates Connection Name by Default

In previous versions, SQL used the database environment string as the connection name if a connection was not specified in a CONNECT statement. Beginning with V7.0, SQL generates a unique default connection name if a connection is not specified in a CONNECT statement.

In previous versions, the following statement created a new connection named 'ATTACH FILENAME PERSONNEL'. In V7.0, SQL generates a unique connection name.

```
SQL> CONNECT TO 'ATTACH FILENAME PERSONNEL';
```

3.2.24 ALTER DATABASE Handles EXTENT Attribute Correctly

In previous versions, the ALTER DATABASE . . . EXTENT IS DISABLED clause was ignored by Oracle Rdb.

In addition, in previous versions, you could not alter the EXTENT attribute for the storage area RDB$SYSTEM using SQL; you had to use Oracle RMU. This restriction is lifted in Oracle Rdb V7.0. Now, you can specify the RDB$SYSTEM storage area in a ALTER DATABASE . . . ALTER STORAGE AREA statement.

Note that the syntax ALTER DATABASE . . . EXTENT IS DISABLED does not alter existing storage area EXTENT attributes. Rather, this syntax provides a default for all newly added storage areas. To change the EXTENT attribute on existing areas, you must use the ALTER DATABASE . . . ALTER STORAGE AREA clause on each area to be changed.

3.2.25 Now Can Create Storage Maps for Tables Containing Data

In previous versions, you could not create a storage map for a table after data had been inserted into the table. A table with no storage map is automatically mapped to the default storage area. To move this table to another area required that the data be unloaded, and the data deleted. Then, the user could have created a storage map to map the table to another area.

This restriction has been lifted in V7.0. You can create a storage area for a table, even if the table contains data. Of course, no storage map can currently exist for this table.

The storage map must be a simple map that references only the default storage area and represents the current (default) mapping for the table. The default storage area is either RDB$SYSTEM or the area name provided by the CREATE DATABASE DEFAULT STORAGE AREA clause.

The storage map may not change thresholds or compression for the table, nor can it use the PLACEMENT VIA INDEX clause. It can contain only one area and cannot be vertically partitioned. The new storage map simply describes the mapping as it exists by default for the table.

After the storage map is created, you can use the ALTER STORAGE MAP statement to reorganize the table as required.

For more information, see the *Oracle Rdb7 Guide to Database Design and Definition* and the *Oracle Rdb7 SQL Reference Manual.*

3.2.26 Views Containing SELECT Literal Now Return Correct Results

In previous versions, a query that selected from a view did not always return the correct results when the following conditions were true:

- SELECT statements within the view returned a constant literal for one of the columns in the view definition.
- The selection predicate (WHERE clause) for the SELECT statement used for the view specified a condition for the column that may have returned a literal value in the view definition.

For example, the following view did not always return correct results:

```
SQL> SHO VIEW VIEW_1
Information for table VIEW_1

Columns for view VIEW_1:
Column Name                     Data Type        Domain
-----------                     ---------        ------
VIEW_COL1                       INTEGER
VIEW_COL2                       INTEGER
VIEW_COL3                       INTEGER
VIEW_COL4                       INTEGER
 Source:
         select
         C1.COL1, C1.COL2, C1.COL3, C1.COL4
         from TABLE_1 C1, TABLE_2 C2, TABLE_3 C3
         where (
             (C1.COL1 = C2.COL1)
             and (C2.COL1 = C3.COL1)
               )
         union all
         select C5.COL1, C5.COL2, C5.COL3, C5.COL4
         from TABLE_1 C4, TABLE_2 C5, TABLE_3 C6,
         where (
             (C4.COL5 = C5.COL5)
             and (C5.COL5 = C6.COL5)
               )
         union
         select C8.COL1, C8.COL2, 0, C8.COL4
-- Note literal for COL3         ^^^
         from TABLE_1 C7, TABLE_2 C8
         where C9.COL10 = C10.COL10
```

Note, in the previous example, that the third select statement returns a literal for the third column.

The following example shows incorrect results that may have been returned by the query. The first row returns the value 5 in VIEW_COL3, which violates the WHERE predicate specified for the SELECT from the view:

```
SQL> SELECT * FROM VIEW_1 WHERE VIEW_COL1=1 AND VIEW_COL3 > 15;
 VIEW_COL1    VIEW_COL2    VIEW_COL3    VIEW_COL4
         1            0            5            0
         1            0           16            0
         1            0           16            0
         1            0           16            0
3 rows selected
SQL>
```

In this situation, Oracle Rdb did not take into account that one of the SELECT statements in the view definition was returning a literal value for one of the columns in the WHERE predicate specified for the view.

This problem has been corrected in Oracle Rdb V7.0.

3.2.27 SELECT DISTINCT from View Now Returns Correct Dbkey

In previous versins, when retrieving the database key (dbkey) of a row from a view that contained a SELECT DISTINCT expression, it was possible to get an incorrect dbkey.

The following example shows a view that returned an incorrect dbkey:

```
CREATE TABLE A (A1 CHAR(1), A2 CHAR(1));
CREATE VIEW VA_WRONG (VA1, VA2) AS SELECT DISTINCT A1,A2 FROM A;

INSERT INTO A VALUES ('B','B');
INSERT INTO A VALUES ('D','D');
INSERT INTO A VALUES ('C','C');

SELECT DBKEY,* FROM VA_WRONG
  WHERE DBKEY = (SELECT DBKEY FROM VA_WRONG WHERE VA1 = 'D');
```

This problem has been corrected in Oracle Rdb V7.0.

3.2.28 Divide by Zero Fault Corrected

In previous versions, it was possible to encounter a divide-by-zero fault even though the query tried to prevent the division by zero, as shown in the following excerpt:

```
WHERE
        (.20 <= (CASE
         WHEN sales_summary.total_sales  = 0 THEN 1.00
         WHEN sales_summary.total_sales <> 0
               THEN sales_summary.total_returns/sales_summary.total_sales
         END)
```

In V6.1, this divide-by-zero fault happened only if the query also included GROUP BY and CASE expressions.

This problem has been corrected in Oracle Rdb V7.0. Oracle Rdb now correctly tests for, and prevents, the division.

3.2.29 Ctrl/Z from SQL HELP No Longer Erases Command Line Recall Buffer

OpenVMS VAX OpenVMS Alpha In previous versions, if you used Ctrl/Z to exit HELP in interactive SQL, the SQL command line recall buffer was erased.

This problem has been corrected in Oracle Rdb V7.0. ♦

3.2.30 Ctrl/Z in Multiscreen Help No Longer Returns RMS-F-EOF Message

OpenVMS VAX OpenVMS Alpha In previous versions, entering Ctrl/Z after the first page in SQL HELP resulted in the following message:

```
%RMS-F-EOF, end of file detected
```

This problem has been corrected in Oracle Rdb V7.0. ♦

3.2.31 Databases Created with MULTITHREADED AREA ADDITIONS Now Correct

In previous versions, the database root file may have been created with incorrect data structures when a database was created using both the MULTITHREADED AREA ADDITIONS clause and the RESERVE STORAGE AREAS clause. Attempts to access storage areas in the resultant database sometimes failed with the following errors:

```
RDB-E-BAD_DPB_CONTENT, invalid database parameters in the database parameter
block (DPB)
RDMS-E-DUPLANAME, area name already used
```

The following example shows this problem:

```
SQL>  CREATE DATABASE FILENAME BIG
cont>    MULTITHREAD AREA ADDITIONS (LIMIT TO 2 AREAS)
cont>    RESERVE 200 STORAGE AREAS
cont>
cont>    CREATE STORAGE AREA RDB$SYSTEM
cont>           FILENAME BIG ALLOCATION IS 200 PAGES
cont>           SNAPSHOT ALLOCATION IS 10 PAGES
cont>    CREATE STORAGE AREA A1 FILENAME A1
cont>           ALLOCATION IS 10 PAGES
cont>           SNAPSHOT ALLOCATION IS 10 PAGES;
cont>
SQL>  ALTER DATABASE FILENAME BIG
cont>    ADD STORAGE AREA M1 FILENAME M1
cont>    ALLOCATION IS 10 PAGES SNAPSHOT ALLOCATION IS 10 PAGES;
cont>
SQL> SHOW STORAGE AREA M1;
No Storage Areas Found

SQL> ALTER DATABASE FILENAME BIG ADD STORAGE AREA M1 FILENAME M1;
%RDB-E-BAD_DPB_CONTENT, invalid database parameters in the database
 parameter block (DPB)
-RDMS-E-DUPLANAME, area name M1 already used
```

The workaround was to avoid using the MULTITHREADED AREA ADDITIONS clause when creating a database.

This problem has been corrected in Oracle Rdb V7.0.

3.2.32 EXPORT and IMPORT Statements Correctly Associate Constraints on Multischema Databases

In previous versions, when you exported and imported a multischema database, constraints were associated with the wrong schema because the catalog and schema information were not correctly exported.

For example, the export and import operations changed the constraint definition from the first example to the second:

```
Original definition:
                Table constraints for MY_CATALOG.SCHEMA1.TABLE1:
                MY_CATALOG.SCHEMA1.COL1_FOREIGN
Definition after export and import:
                Table constraints for MY_CATALOG.SCHEMA1.TABLE1:
                MY_CATALOG.SCHEMA2.COL1_FOREIGN
                             |-> Wrong schema
```

As a result, you could add the same constraint after the import operation without generating an error message.

This problem has been corrected in V7.0. Note that because the catalog and schema information were not correctly exported, you must export the multischema database again to import it correctly.

3.2.33 Behavior of Global Buffering Using IMPORT Corrected

In previous versions, the IMPORT statement did not correctly import global buffering attributes when global buffers were disabled. This problem has been corrected in V7.0, as shown in the following example:

```
SQL> CREATE DATA FILE BUFFER_TEST
cont> GLOBAL BUFFERS ARE DISABLED
cont> (NUMBER IS 222, USE LIMIT IS 22);
SQL> EXPORT DATA FILE BUFFER_TEST INTO NEW_BUFFER_TEST;
SQL> DISCONNECT ALL;
SQL> -- IMPORT now shows global buffers as disabled, and the correct number
SQL> -- and user limit
SQL> IMPORT DATA FROM NEW_BUFFER_TEST FILE BUFFER_TEST;
Exported by Oracle Rdb V7.0-00 Import/Export utility
A component of SQL V7.0-00
Previous name was buffer_test
It was logically exported on 22-JAN-1996 19:04
   .
   .
   .
Adjustable Lock Granularity is Enabled Count is 3
Database global buffering is DISABLED
Database number of global buffers is 222
Number of global buffers per user is 22
Database global buffer page transfer is via DISK
Journal fast commit is DISABLED
Journal fast commit checkpoint interval is 0 blocks
Journal fast commit checkpoint time is 0 seconds
Commit to journal optimization is Disabled
   .
   .
   .
```

3.2.34 EXPORT and IMPORT Statements Handle Invalidated Outlines Correctly

In previous versions, Oracle Rdb exported and imported invalidated outlines. Then, the IMPORT statement failed with errors such as the following:

```
%SQL-F-NOOUTLRES, Unable to IMPORT outline DEMO
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-TABNOTDEF, relation T1 is not defined in database
```

This problem has been corrected in V7.0. The IMPORT statement no longer fails when you export and import invalidated outlines. Oracle Rdb exports invalidated outlines and marks them as invalid during the import. In addition, Oracle Rdb displays a message indicating the outline is marked as invalid and is not imported.

The following example illustrates this behavior:

```
SQL> CREATE DATABASE FILENAME OUTLINE_TEST;
SQL> CREATE TABLE T1 (F1 CHAR(10));
SQL> CREATE INDEX I1 ON T1 (F1);
SQL> COMMIT;
SQL> CREATE OUTLINE DEMO
cont>   ID '7900E969D7D406FD9DC09AEBBB9B32F2'
cont>   MODE 0
cont>   AS (QUERY (SUBQUERY (t1 0 ACCESS PATH INDEX i1) ) )
cont>   COMPLIANCE OPTIONAL;
SQL> COMMIT;
SQL>
SQL> DROP TABLE T1;
SQL> COMMIT;
SQL> -- The outline is now marked invalid.
SQL> SHOW OUTLINE DEMO;
     DEMO
    Object has been marked INVALID
CREATE OUTLINE DEMO
ID '7900E969D7D406FD9DC09AEBBB9B32F2'
MODE 0
AS (
  QUERY (
    SUBQUERY (
      t1 0    ACCESS PATH INDEX       i1
      )
    )
  )
COMPLIANCE OPTIONAL     ;
SQL> DISCONNECT ALL;
SQL> --
SQL> -- Export the database.
SQL> EXPORT DATA FILE OUTLINE_TEST INTO DEMO;
SQL> -- Import no longer fails.  A message is displayed indicating the
SQL> -- outline is marked as invalid.
SQL> IMPORT DATA FROM DEMO FILENAME DEMO;
Exported by Oracle Rdb V7.0-00 Import/Export utility
A component of SQL V7.0-00
Previous name was outline_test
It was logically exported on 22-JAN-1996 19:35
   .
   .
   .
IMPORTing STORAGE AREA: RDB$SYSTEM
Outline DEMO is marked as invalid, and is not imported
```

3.2.35 Export and Import Files Now Can Use Any Extension

In previous versions, SQL EXPORT and IMPORT statements only worked using the file extension .rbr. Attempts to export to another extension resulted in an export file with the .rbr extension. Attempts to import from an interchange file with an extension other than .rbr resulted in the attempt to open the interchange file with an .rbr extension.

This problem has been corrected in Oracle Rdb V7.0.

3.2.36 Now Can Import Interchange Files with Uppercase Extensions on Digital UNIX

Digital UNIX

In V6.1 on Digital UNIX, SQL did not import an interchange file if any of the letters in the filename extension were in upper case. The workaround was to rename the interchange file to one with a lowercase extension.

This problem has been corrected in Oracle Rdb V7.0. ♦

3.2.37 IMPORT Statement No Longer Exceeds Memory

In previous versions, attempts to use the SQL IMPORT statement to import a database with many stored procedures could have failed with a virtual memory exceeded error.

The problem was due to a memory leak (for example, unreleased virtual memory) during the CREATE MODULE statement which is executed by IMPORT when building the new database. As each routine was defined, some of the data structures allocated in dynamic memory were not released. As a result, the amount of virtual memory available to the process dwindled.

In some cases, it was possible to alter the process quotas (page file quota) or system parameters (virtual page count) to work around this problem.

This problem has been corrected in Oracle Rdb V7.0.

3.2.38 EXPORT and IMPORT Statement Problems with Procedures and Functions in ANSI Databases Are Fixed

In previous versions, users experienced various procedure and function problems with the SQL EXPORT or IMPORT statements when using ANSI-style privileges databases. Often, EXPORT or IMPORT consumed all memory available to the process files. Other problems included ACLs attached to the wrong objects.

This problem has been corrected in Oracle Rdb V7.0.

3.2.39 SQL92 Intermediate Level UNIQUE Constraint Available

Oracle Rdb now provides a UNIQUE constraint that is compliant with the intermediate level of the SQL92 standard. This type of constraint excludes columns that are NULL from the UNIQUE comparison, effectively allowing sets of columns to be UNIQUE or NULL.

When you set the SQL dialect to SQL89, MIA, ORACLE LEVEL1, or SQL92, SQL uses this type of constraint by default. (The default dialect is currently SQLV40.) Oracle Rdb recommends that you set the dialect to SQL92 (or one of the listed dialects) before using the CREATE or ALTER TABLE statements to add UNIQUE constraints to tables.

Note

The new UNIQUE semantics will be used at run time under any selected dialect. That is, the table must be created or altered under the listed dialects to have the new style of unique enabled.

In addition to conforming to the SQL92 Intermediate Level standard, the new UNIQUE constraint implementation provides improved performance for single row inserts. This is made possible by eliminating checks for NULL values from the selection expression, thus simplifying the optimization for unique checking.

The following example shows a comparison of the old and new optimizer strategies. In this example, the UNIQUE constraint, UNIQUE_A, and an index on column A are used to check for uniqueness during an INSERT statement. Note that the optimizer chooses a full range search of the index (that is, [0:0]).

```
~S: Constraint "UNIQUE_A" evaluated
Cross block of 2 entries
  Cross block entry 1
    Conjunct       Firstn  Get     Retrieval by DBK of relation T_UNIQUE
  Cross block entry 2
    Conjunct       Aggregate-F2    Conjunct
    Index only retrieval of relation T_UNIQUE
      Index name  T_UNIQUE_INDEX_A [0:0]
```

With the simplified UNIQUE constraint, UNIQUE_B, the optimizer uses a direct lookup of the index (that is, [1:1]), which reduces the I/O the index needs to perform the constraint evaluation.

```
~S: Constraint "UNIQUE_B" evaluated
Cross block of 2 entries
  Cross block entry 1
    Conjunct       Firstn  Get     Retrieval by DBK of relation T_UNIQUE
  Cross block entry 2
    Conjunct       Aggregate-F2    Index only retrieval of relation T_UNIQUE
      Index name  T_UNIQUE_INDEX_B [1:1]
```

In previous versions, the UNIQUE constraint restricted columns to a single NULL value. To retain this behavior, use the SET DIALECT 'SQLV40' statement before creating new tables or altering existing tables to add UNIQUE constraints.

Tables created in previous versions of Oracle Rdb still perform as in previous versions. Constraints defined through interfaces such as RDO or the CDD/Repository retain the older style UNIQUE constraint. Future versions of the Oracle CDD/Repository may implement the new UNIQUE constraint. Databases exported and imported using the SQL EXPORT and IMPORT statements retain the UNIQUE constraint as it is defined in the database, regardless of the dialect setting.

Note

RMU Extract Item=Table will not distinguish between the old and new UNIQUE constraints in this release of Oracle Rdb. The generated SQL script must be modified to establish the appropriate dialect before using it to create a database.

Because this new style of UNIQUE constraint is a relaxation of the UNIQUE rules, you can drop the old style UNIQUE constraint and redefine the constraint under the SQL92 dialect.

Note that this meaning of UNIQUE (that is, excluding NULL from the uniqueness test) does not apply to the UNIQUE index. A UNIQUE index does not allow duplicate entries for NULL. If a UNIQUE index is currently defined which assists the UNIQUE constraint optimization, you may wish to drop the index and make it a non-UNIQUE index so that multiple NULL values can be stored. The UNIQUE constraint will enforce the uniqueness of the data.

You can use the SQL SHOW TABLE command to determine which type of UNIQUE constraint is in use. For example, the first example is a UNIQUE constraint created when the default dialect was used (SQLV40). A new description follows the "Unique constraint" text, explaining the interpretation of null values.

```
SQL> SHOW TABLE (CONSTRAINT) T_UNIQUE
Information for table T_UNIQUE

Table constraints for T_UNIQUE:
T_UNIQUE_UNIQUE_B_A
 Unique constraint
     Null values are considered the same
 Table constraint for T_UNIQUE
 Evaluated on UPDATE, NOT DEFERRABLE
 Source:
        UNIQUE (b,a)
   .
   .
   .
```

This second example is a UNIQUE constraint created when the dialect was set to 'SQL92', and the description here indicates that all null values are considered distinct.

```
SQL> SHOW TABLE (CONSTRAINT) T_UNIQUE2
Information for table T_UNIQUE2

Table constraints for T_UNIQUE2:
T_UNIQUE2_UNIQUE_B_A
 Unique constraint
     Null values are considered distinct
 Table constraint for T_UNIQUE2
 Evaluated on UPDATE, NOT DEFERRABLE
 Source:
        UNIQUE (b,a)
   .
   .
   .
```

As a side effect of this change, Oracle Rdb recognizes a larger class of CHECK constraints as being uniqueness checks. As a result, these constraints are no longer executed when a DELETE statement is executed for the table, because a delete operation does not affect the uniqueness of the remaining rows. The following example shows a CHECK constraint with this characteristic:

```
SQL> create table T_USER_UNIQUE_NEW (
cont>     a integer,
cont>     b integer,
cont>     constraint unique_ab_new
cont>         check ((select count(*)
cont>                 from T_USER_UNIQUE_NEW t2
cont>                 where t2.a = T_USER_UNIQUE_NEW.a and
cont>                       t2.b = T_USER_UNIQUE_NEW.b) <= 1)
cont>             not deferrable
cont>     );
```

In previous versions, Oracle Rdb recognized only equality with 1 as a uniqueness constraint. In this example, a comparison of less than or equal to 1 also qualifies as a uniqueness constraint.

3.2.40 Constraints No Longer Fail When New Column Is Created Using DEFAULT

In previous versions, when you added a new column using a DEFAULT clause, Oracle Rdb evaluated any constraints before it applied the DEFAULT clause to pre-existing rows. As a result, a constraint violation was detected when one did not really exist.

The following script example shows a constraint that was evaluated too early:

```
SQL> CREATE DOMAIN DECUS_DOM INTEGER DEFAULT 99;
SQL>
SQL> ALTER TABLE EMPLOYEES
cont>     ADD COLUMN DECUS_COL  DECUS_DOM
cont>     CONSTRAINT DECUS_CONS NOT NULL NOT DEFERRABLE;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-INTEG_FAIL, violation of constraint DECUS_CONS caused operation to fail
-RDB-F-ON_DB, on database DISK1:[TEST.DATABASES]MF_PERSONNEL.RDB;1
```

The workaround was to change the ALTER TABLE statement to be two separate statements, as shown in the following example:

```
SQL> ALTER TABLE EMPLOYEES
cont>     ADD COLUMN DECUS_COL  DECUS_DOM;
SQL> ALTER TABLE EMPLOYEES
cont>     ALTER COLUMN DECUS_COL
cont>         CONSTRAINT DECUS_CONS NOT NULL NOT DEFERRABLE;
```

This problem is corrected in Oracle Rdb V7.0. Now, Oracle Rdb evaluates constraints after it applies the default value.

3.2.41 New Behavior for Domain Check Constraints and NULL

The SQL domain CHECK constraint is implemented using the VALID IF clause (available in RDO and CDD/Repository). However, the VALID IF clause implements different semantics from an SQL column CHECK constraint as defined by the ANSI/ISO SQL standard. The difference is in the handling of the check expression when it is evaluated as NULL.

For example, the column CHECK constraint `COL < 0` fails if the value of COL exceeds 0. However, a null value does not cause a column constraint failure. On the other hand, in previous versions, a domain CHECK constraint caused a failure when the expression evaluated to NULL.

For all new or modified domain CHECK constraints, Oracle Rdb V7.0 now applies the same semantics for NULL as for column CHECK constraints. Any domain CHECK constraints created by previous versions of Oracle Rdb continue to exhibit incorrect (or at least inconsistent) handling of NULL when compared to column CHECK constraints.

To inherit the new behavior, you must redefine the domain CHECK constraint for all domains in the database. The following example shows an ALTER DOMAIN statement that redefines the CHECK constraint:

```
SQL> ALTER DOMAIN salary CHECK (VALUE > 0.00) NOT DEFERRABLE;
```

The ALTER DOMAIN statement causes the new definition to be propagated to all columns based on the domain.

Note

The behavior of CDD or RDO constraints is unchanged. The constraint must be defined through the SQL interface to inherit the semantics described by this release note.

In previous versions, the workaround was to define the CHECK constraint as: CHECK (value is NULL or check-expression).

3.2.42 RDB$MESSAGE_VECTOR Psect Size Corrected for OpenVMS Alpha

OpenVMS Alpha

In previous versions, using the RDB$MESSAGE_VECTOR psect with the SQL precompiler or SQL module processor (especially when you used INCLUDE SQLCA), could have resulted in 160 byte psect lengths instead of the correct 80 byte psect length on OpenVMS Alpha systems. This problem probably was not noticed by most users, except those who used multiple shared images with multiple versions of Oracle Rdb.

The following sample C program shows the problem in a SQL precompiled program:

```
exec sql declare alias filename mf_personnel;

main()
{    long SQLCODE;
     long x;

     exec sql create outline out1 from (select last_name from employees);
     printf("SQLCODE: %d\n",SQLCODE);
     SQL$SIGNAL();
     exec sql create outline out2 from (select employee_id from job_history);
     printf("SQLCODE: %d\n",SQLCODE);
     SQL$SIGNAL();

     exec sql commit;
}
```

This problem has been corrected in V7.0. ♦

3.2.43 CANTSNAP Errors Stop Occurring After Multiple Re-Ready Requests

When an exclusive transaction runs concurrently with a read-only transaction, the exclusive transaction does not write to the snapshot file, and the read-only transaction can therefore get a inconsistent view of the modified records from the snapshot file. This situation is normally prevented by the CANTSNAP error being returned to the read-only transaction.

In previous versions, however, a problem that caused the CANTSNAP error to only get issued once per table partition to the read-only transaction had been identified. If the read-only transaction retried its query multiple times, the query eventually succeeded and resulted in a subsequent bugcheck.

The following example shows that the first attempt at reading from JOBS was correctly rejected with the CANTSNAP error, but the second attempt proceeded and then failed with the bugcheck:

```
 Session 1                       Session 2
 -----------------------------   ---------------------------------------------
SQL> SET TRANS READ ONLY;
                                 SQL> SET TRANS READ WRITE RESERVING JOBS
                                 cont> FOR EXCLUSIVE WRITE;

                                 SQL> UPDATE JOBS SET MINIMUM_SALARY = 100000
                                 cont> WHERE JOB_CODE = 'APGM';

                                 SQL> COMMIT;

SQL> SET TRANS READ ONLY;

SQL> SELE * FROM JOBS;
%RDB-E-LOCK_CONFLICT, request failed due to locked resource
-RDMS-F-CANTSNAP, can't ready storage area
DBS1:[JOE_USER.RDBLAB.MFDB_V60]JOBS.RDA;1 for snapshots
```

```
SQL> SELE * FROM JOBS;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DBS1:[JOEUSR]RDSBUGCHK.DMP;4
```

Interestingly, if you did the same experiment using the EMPLOYEE table, it took three retries in the read-only transaction before getting the bugcheck, as shown in the following example:

```
 Session 1                       Session 2
 ----------------------------    --------------------------------------------
SQL> SET TRANS READ ONLY;
                                 SQL> SET TRANS READ WRITE RESERVING EMPLOYEES
                                 cont> FOR EXCLUSIVE WRITE;

                                 SQL> UPDATE EMPLOYEES SET CITY='Nice'
                                 cont> WHERE EMPLOYEE_ID = '00164';

                                 SQL> COMMIT;

SQL> SELE * FROM EMPLOYEES WHERE EMPLOYEE_ID = '00164';
%RDB-E-LOCK_CONFLICT, request failed due to locked resource
-RDMS-F-CANTSNAP, can't ready storage area
DBS1:[JOE_USER.RDBLAB.MFDB_V60]EMPIDS_LOW.RDA;1 for snapshots

SQL> SELE * FROM EMPLOYEES WHERE EMPLOYEE_ID = '00164';
%RDB-E-LOCK_CONFLICT, request failed due to locked resource
-RDMS-F-CANTSNAP, can't ready storage area
DBS1:[JOE_USER.RDBLAB.MFDB_V60]EMPIDS_LOW.RDA;1 for snapshots

SQL> SELE * FROM EMPLOYEES WHERE EMPLOYEE_ID = '00164';
%RDB-E-LOCK_CONFLICT, request failed due to locked resource
-RDMS-F-CANTSNAP, can't ready storage area
DBS1:[JOE_USER.RDBLAB.MFDB_V60]EMPIDS_LOW.RDA;1 for snapshots

SQL> SELE * FROM EMPLOYEES WHERE EMPLOYEE_ID = '00164';
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DBS1:[JOE_USER]RDSBUGCHK.DMP;3
%COSI-F-BUGCHECK, internal consistency failure
```

This occurred because the EMPLOYEE table is partitioned using three logical areas and each ready failed in turn.

There was no workaround to this problem.

This problem has been corrected in Oracle Rdb V7.0. The CANTSNAP error is now correctly returned all of the time.

3.2.44 Multistatement Procedure Using a Labeled FOR Statement No Longer Bugchecks

OpenVMS VAX OpenVMS Alpha

In previous versions, using a labeled FOR statement inside a loop in a multistatement procedure sometimes resulted in a SQL bugcheck dump in SQL$$CREATE_STMT. The bugcheck contained the text `SQL$SEM - 0`.

The following example shows the problem in interactive SQL:

```
SQL> ATTACH 'FILENAME mf_personnel';
SQL> DECLARE :i BIGINT;
SQL> BEGIN
cont>     WHILE :i > 0
cont>     LOOP
cont>   po_loop:
```

```
cont>   FOR :aa1 AS EACH ROW OF TABLE CURSOR tc1 FOR
cont>     SELECT EMPLOYEE_ID FROM EMPLOYEES DO
cont>       SET :i = 0;
cont>   END FOR;
cont>     END LOOP;
cont> END;
%SQL-I-BUGCHKDMP, generating bugcheck dump file USD04:[MOY]SQLBUGCHK.DMP;
%SQL-F-BUGCHK, There has been a fatal error. Please submit a software
performance report. SQL$SEM - 0
```

This problem has been corrected in V7.0.

3.2.45 Dynamic SQL and TRIM No Longer Result in Access Violation

In previous versions, use of TRIM({BOTH, TRAILING, or LEADING} FROM {source}) could have resulted in access violations when used with dynamic SQL.

The following precompiled SQL program reproduced this problem:

```
#include <string.h>
#include <stdio.h>
main() {int      SQLCODE;
        char     cmd[128];
        strcpy(cmd,"attach 'filename home1:mf_personnel'");
        EXEC SQL execute immediate :cmd;
        strcpy(cmd,"select trim(trailing from last_name) from employees");
        EXEC SQL prepare dyn_statement from :cmd;
        if (SQLCODE != 0)
            {   printf("Error on prepare");
                sql$signal(); } }
```

The workaround was to always specify the trim character.

This problem has been corrected in Oracle Rdb V7.0.

3.2.46 Intervals in Views No Longer Loop

In previous versions, when you selected an interval from a view, the process looped and never returned. For example, because DATE_VIEW is a view, the following query looped and never returned a value:

```
SQL> SELECT * FROM DATE_VIEW
cont>       WHERE DATE_DIFF=INTERVAL '0' YEARS;
```

This problem has been corrected in Oracle Rdb V7.0.

3.2.47 Using COALESCE with Aggregate Functions Now Returns Correct Results

In previous versions, when you used a conditional expression and the COALESCE function with aggregate functions, Oracle Rdb could have returned incorrect results.

The following example shows a query that returned one row incorrectly:

```
SQL> select coalesce(sum(salary_amount),0) from salary_history
cont> where employee_id = '00160';
0 rows selected
```

This problem has been corrected in Oracle Rdb V7.0.

3.2.48 SELECT . . . LIKE with Host Variable No Longer Fails

In previous versions, when varying the length of the string in host variables to shorter strings, the SELECT statement often failed to retrieve data. The following shows an example of this type of SELECT statement:

```
SELECT {column} FROM {table} WHERE {column2} LIKE :{hostfield} ;
```

Workarounds included such drastic remedies as exiting the program and rerunning it.

This problem has been corrected in Oracle Rdb V7.0.

3.2.49 SQL No Longer Bugchecks at SQL$$SET_TERM_CHARS + xxxxxx

In previous versions, SQL bugchecked at SQL$$SET_TERM_CHARS + xxxxxx when long query headers were used. This occurred when the length of the query header plus the length of the indent (generated by Interactive SQL) was greater than the length of the line.

The following shows an example that generated the problem:

```
set language ENGLISH;
set quoting rules 'SQL92';
CREATE DATABASE FILENAME MIKE;

create domain MTAGIO     BIGINT (2)
    query header is 'Mtt total d''agios/escompte';
    comment on domain MTAGIO is
        'Montant total d''agios/escompte';

create domain MTFCOF     BIGINT (2)
    query header is 'Mtt total de frais Coface';
    comment on domain MTFCOF is 'Montant total de frais Coface';

create domain MTFHPV
    BIGINT (2)
    query header is 'montant frais HPV';
    comment on domain MTFHPV is 'montant frais HPV';

   .
   .
   .
create table MVTSTA (
   .
   .
   .
    MSAGIO MTAGIO
      query header is
      'total agios/escpte pour la ligne, positif si création,'
      ' négatif si annulation'
    MSFCOF MTFCOF
      query header is
      'total frais coface pour la ligne, signé négativement si annulation',
    MSFHPV MTFHPV
      query header is
      'total frais hpv pour la ligne, signé négativement si annulation',
      'qté en unité de prix, signée négativement si annulation',
   .
   .
   .
    TXMANE TXMANE);
comment on table MVTSTA is
'Mouvements quotidiens pour les éditions de suivi des commandes';


insert into MVTSTA (MSAGIO) values (1234);
select * from MVTSTA;
select MSAGIO,MSAGIO,MSAGIO from MVTSTA;
```

The workaround was to use shorter query names or column names.

This problem has been corrected in Oracle Rdb V7.0.

3.2.50 Bugcheck Error in SHOW TRANSACTION Fixed

In previous versions, the SQL SHOW TRANSACTION statement could have resulted in a bugcheck with an exception in SQL$$OUTPUT_MSG with an additional message: `SQL$EXESHO - 9`. The SHOW TRANSACTION statement displays information about the characteristics of the current transaction. The problem occurred when there was one disconnection in a multiple connection.

The following example shows the coding in which this problem occurred:

```
SQL> ATTACH 'FILE PERSONNEL';
SQL> CONNECT TO 'FILE PERSONNEL' AS 'C2';
SQL> DISCONNECT DEFAULT;
SQL> SELECT COUNT(*) FROM RDB$RELATIONS;
SQL> COMMIT;
SQL> SHOW TRANSACTION
```

This problem has been fixed for Oracle Rdb V7.0.

3.2.51 ALTER TABLE Column Deletion Errors Fixed

In previous versions, you could have received an error if you tried to update a column in an existing table definition after using the SQL statement ALTER TABLE with the DROP COLUMN clause. The error occurred when the column to be deleted was of LIST OF BYTE VARYING data type.

The following example shows the error that occurred:

```
SQL> ALTER TABLE RESUMES DROP COLUMN RESUME;
SQL> COMMIT;
SQL> ALTER TABLE RESUMES ADD COLUMN JUNK CHAR(5);
SQL> UPDATE RESUMES SET JUNK = 'abcd' WHERE EMPLOYEE_ID='00165';
%RDB-E-NO_RECORD,
-RDMS-F-NODBK
```

This problem has been corrected in Oracle Rdb V7.0.

3.2.52 Using Dynamic Statement Names No Longer Causes Memory Leaks

In previous versions, using dynamic statement names in dynamic SQL sometimes caused memory leaks when different statement names were used.

An example of the problem is a PREPARE statement with dynamic statement names, such as the following:

```
EXEC SQL PREPARE :STMT_NAME FROM :STMT
```

A program containing this statement may not have released memory when you used a RELEASE :STMT_NAME statement, causing memory growth problems in programs that used many different statement names. This problem affected dynamic SQL using both precompiled SQL and SQL module language interfaces.

This problem has been corrected in Oracle Rdb V7.0.

3.2.53 Now Can Modify Data Types When Constraints Are Defined

Since Oracle Rdb Version 4.2, the SQL ALTER statement failed when it tried to update the data type of a column and a constraint existed that referred to the column. Oracle Rdb did not permit columns to be updated because the altered column might contain a value that would violate the constraint. The ALTER statement failed and returned an error message as shown in the following example:

```
SQL> CREATE DOMAIN t1 CHAR (6);
SQL> CREATE TABLE t (f t1
cont>                CONSTRAINT f_constraint
cont>                CHECK (f > 'abc') DEFERRABLE);
   .
   .
   .
SQL> ALTER DOMAIN T1 CHAR (3);
%SQL-W-CHR_TOO_SHO,  Character length of domain T1 is too short
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-FLDINCON, field F is referenced in constraint F_CONSTRAINT
-RDMS-F-FLDNOTCHG, field F has not been changed
```

Beginning with V7.0, Oracle Rdb checks the needed constraints and permits the ALTER TABLE or ALTER DOMAIN statement to succeed if no constraints are violated. For example, the following ALTER DOMAIN statement successfully updates the column:

```
SQL> INSERT INTO t VALUE ('abd');
SQL> SELECT f FROM t;
abd

SQL> ALTER DOMAIN t1 CHAR (3);
%SQL-W-CHR_TOO_SHO,  Character length of domain T1 is too short
```

Note that the ALTER TABLE or DOMAIN statements now return a warning message, even when the column has been updated successfully. However, if the ALTER statement tries to perform an update that will result in a constraint violation, the statement fails and returns an error message. For example:

```
SQL> ALTER DOMAIN t1 CHAR (2);
%SQL-W-CHR_TOO_SHO,  Character length of domain T1 is too short
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-INTEG_FAIL, violation of constraint F_CONSTRAINT caused operation to
fail
-RDB-F-ON_DB, on database personnel
```

3.2.54 ALTER TABLE No Longer Causes BEFORE or AFTER UPDATE Triggers to Execute Unexpectedly

In previous versions, the following ALTER TABLE statements may have caused BEFORE or AFTER UPDATE triggers to fire unexpectedly:

- ALTER TABLE ADD COLUMN: When the column had an explicit default value or inherited a default value from a domain in which the default value was not NULL
- ALTER TABLE ALTER COLUMN: When either RDMS$BIND_UPDATE_CHANGED_RELATION or RDMS$BIND_VALIDATE_CHANGE_FIELD logical names equated to the value "1" and an ALTER TABLE statement modified a collating sequence or data type of a column
- ALTER TABLE DROP COLUMN: When the data type of the dropped column was LIST OF BYTE VARYING

Each of these ALTER TABLE statements required a query to execute to update the table rows. For example, an update to the table rows might include propagating the DEFAULT value, validating the data type change, or cascading the delete of the LIST OF BYTE VARYING data. These updates caused the BEFORE or AFTER UPDATE triggers to execute unexpectedly.

This problem has been corrected in Oracle Rdb V7.0. The triggers no longer execute during ALTER TABLE. This correction causes the full table update to have a more efficient execution strategy.

3.2.55 Simple CASE Expressions Are Evaluated Correctly

In Oracle Rdb V6.0, SQL introduced two forms of the CASE expression: the simple and the searched case expression. The simple case expression is a short-hand that allows the programmer to specify the matching value expression once and then list the equality matching expressions. For example:

```
CASE (SELECT COUNT(*) FROM EMPLOYEES)
    WHEN 0 THEN 'DATABASE EMPTY'
    WHEN 100 THEN 'DATABASE HAS EXPECTED NUMBER OF EMPLOYEES'
    ELSE 'DATABASE HAS EMPLOYEES'
END
```

This type of CASE expression calculates the value expression once and then compares it to each WHEN expression. When the primary expression is complex (such as the subquery in the example) this format provides a slight performance benefit.

Unfortunately, in the following situations, when the evaluation of the primary expression was performed at the wrong time, it could have produced incorrect results:

- When the computed column was sorted using the ORDER BY clause

  Oracle Rdb performed the evaluation after the sorting, generating an incorrect sort order.
- When the expression was used in a subquery

  The optimizer may have chosen to reorder the evaluation, and again evaluate the CASE expression with the wrong value.

The searched CASE expression did not suffer from these problems and could have been used as an alternative.

This problem has been corrected in Oracle Rdb V7.0.

3.2.56 New Warning Message Is Generated for Redundant Column References

Oracle Rdb Version 7.0 now generates a warning message when it finds redundant column references in an index definition. A message is generated because, when a column is referenced more than once in an SQL index definition, the reference adds no useful information and increases the size of the index key. Thus, redundant column references use more disk space to store the index.

The following example shows the new warning message:

```
SQL> CREATE INDEX DUP_COL ON EMPLOYEES (EMPLOYEE_ID, LAST_NAME, EMPLOYEE_ID);
%RDB-W-META_WARN, metadata successfully updated with the reported warning
-RDMS-W-IDXDUPCOL, index definition contains a redundant reference to a column
```

Note that the index is created even though the warning message is generated.

3.2.57 Value Restriction Removed for Indexes and Storage Maps

In previous versions, Oracle Rdb limited the values in storage maps and index definitions. This affected the following SQL statements:

- CREATE INDEX
- ALTER INDEX
- CREATE STORAGE MAP
- ALTER STORAGE MAP

If the columns used for partitioning had numeric data types and the values specified for partitioning were text strings, the actual data partitioning was counterintuitive, provided incorrect results, or resulted in bugchecks.

The following example shows the coding that might cause this problem:

```
CREATE INDEX X1 ON T1 (F1 ASC MAP VALUES -1 to 2) TYPE HASHED
       STORE USING (F1)
             IN a1 WITH LIMIT OF ('-1')
             IN a2 WITH LIMIT OF  ('0')
             IN a3 WITH LIMIT OF  ('1') ;
```

This problem has been corrected in Oracle Rdb V7.0.

3.2.58 ACCVIO or Memory Consumption Loop Problems Using SQL Module Language and Connections Are Fixed

In Oracle Rdb V6.0 and V6.1, programs that used SQL module language and connections could have terminated with access violations or with memory allocation failure errors. This problem was due to ACCVIO or memory consumption loop problems that caused blocks of memory to be prematurely deallocated.

The following example shows how this problem might have occurred. One Fortran program and three SQL module language sources are compiled and linked together. The SQL modules should have been compiled with the connect qualifier.

The following example shows the Fortran program:

```
        PROGRAM TEST_MULTI_CONNECT

        IMPLICIT        NONE
!
!       Local variables.
        CHARACTER*128   MSG_BUFFER
        INTEGER*2       MSG_LENGTH
        INTEGER*4       SQLCODE
        CHARACTER*32    CONNECT_NAME
!
        CONNECT_NAME = 'CONNECTION_1'
!
!       Connect to the TDF database.
        CALL TDF_CONNECT (SQLCODE, CONNECT_NAME)
        IF (SQLCODE .LT. 0) GOTO 34
!
!       Start a read/write transaction.
        CALL TDF_SET_TRANS_RW (SQLCODE)
        IF (SQLCODE .LT. 0) GOTO 34
!
!       Commit.
        CALL TDF_COMMIT (SQLCODE)
        IF (SQLCODE .LT. 0) GOTO 34
```

```
!
!       Start a read-only transaction.
        CALL TDF_SET_TRANS_RO (SQLCODE)
        IF (SQLCODE .LT. 0) GOTO 34
!
!       Rollback the transaction.
        CALL TDF_ROLLBACK (SQLCODE)
        IF (SQLCODE .LT. 0) GOTO 34
!
!       Disconnect from TDF_DB.
        CALL DISCONNECT (SQLCODE, CONNECT_NAME)
        IF (SQLCODE .LT. 0) GOTO 34
!
!
!       CS database
!
        CONNECT_NAME = 'CONNECTION_2'
!
!       Connect to the CS database.
        CALL CS_CONNECT (SQLCODE, CONNECT_NAME)
        IF (SQLCODE .LT. 0) GOTO 34
!
!       Start a read/write transaction.
        CALL TDF_SET_TRANS_RW (SQLCODE)
        IF (SQLCODE .LT. 0) GOTO 34
!
!       Commit the database transaction.
        CALL TDF_COMMIT (SQLCODE)
        IF (SQLCODE .LT. 0) GOTO 34
!
!       Disconnect from CS_DB.
        CALL DISCONNECT (SQLCODE, CONNECT_NAME)
        IF (SQLCODE .LT. 0) GOTO 34
!
!
!       More TDF database
!
        CONNECT_NAME = 'CONNECTION_3'
!
!       Connect to the TDF database.
        CALL TDF_CONNECT (SQLCODE, CONNECT_NAME)
        IF (SQLCODE .LT. 0) GOTO 34
!
!       Start a read/write transaction.
        CALL TDF_SET_TRANS_RW (SQLCODE)
        IF (SQLCODE .LT. 0) GOTO 34
!
!       Commit the database transaction.
        CALL TDF_COMMIT (SQLCODE)
        IF (SQLCODE .LT. 0) GOTO 34
!
!       Disconnect from TDF_DB.
        CALL DISCONNECT (SQLCODE, CONNECT_NAME)
        IF (SQLCODE .LT. 0) GOTO 34
!
!       Done.
        STOP
!
!
!       Error handling.
!
34      CALL SQL$GET_ERROR_TEXT (MSG_BUFFER, MSG_LENGTH)
        TYPE *, MSG_BUFFER (:MSG_LENGTH)
        CALL TDF_ROLLBACK (SQLCODE)

        CALL TDF_ROLLBACK (SQLCODE)
        STOP
        END
```

The following example shows the first SQL module language program:

```
-- Define header information.
MODULE          TDF_CONNECT               -- Module name
LANGUAGE        FORTRAN                   -- Language of calling program
AUTHORIZATION   RDB$DBHANDLE              -- Default authorization identifier
PARAMETER       COLONS                    -- Parameters are prefixed by colons
--
-- Attach to the database.
DECLARE GLOBAL TDF ALIAS FILENAME 'TDF_DB'
--
-- Establish a TDF database connection.
PROCEDURE TDF_CONNECT
        (SQLCODE,
        :CONNECT_NAME CHAR(32));
        CONNECT TO 'ALIAS TDF' AS :CONNECT_NAME;
```

The following example shows the second SQL module language program:

```
-- Define header information.
MODULE          CS_CONNECT                -- Module name
LANGUAGE        FORTRAN                   -- Language of calling program
AUTHORIZATION   RDB$DBHANDLE              -- Default authorization identifier
PARAMETER       COLONS                    -- Parameters are prefixed by colons
--
-- Attach to the database.
DECLARE GLOBAL CS ALIAS FILENAME 'CS_DB'
--
-- Establish a CS database connection.
PROCEDURE CS_CONNECT
        (SQLCODE,
        :CONNECT_NAME CHAR(32));
        CONNECT TO 'ALIAS CS' AS :CONNECT_NAME;
```

The following example shows the third SQL module language program:

```
-- Define header information.
MODULE          GENERAL_CONNECT           -- Module name
LANGUAGE        FORTRAN                   -- Language of calling program
AUTHORIZATION   RDB$DBHANDLE              -- Default authorization identifier
PARAMETER       COLONS                    -- Parameters are prefixed by colons
--
-- Attach to the databases.
DECLARE GLOBAL CS ALIAS FILENAME 'CS_DB'
DECLARE GLOBAL TDF ALIAS FILENAME 'TDF_DB'
--
-- Change to a specific database connection.
PROCEDURE SET_CONNECTION
        (SQLCODE,
        :CONNECT_NAME CHAR(32));
        SET CONNECT :CONNECT_NAME;
--
-- Disconnect from a specific database connection.
PROCEDURE DISCONNECT
        (SQLCODE,
        :CONNECT_NAME CHAR(32));
        DISCONNECT :CONNECT_NAME;
--
-- Set a read-only transaction.
PROCEDURE TDF_SET_TRANS_RO
        (SQLCODE);
        SET TRANSACTION READ ONLY;
```

```
--
-- Set a read/write transaction.
PROCEDURE TDF_SET_TRANS_RW
        (SQLCODE);
        SET TRANSACTION READ WRITE;
--
-- Commit outstanding transaction for the current connection.
PROCEDURE TDF_COMMIT
        (SQLCODE);
        COMMIT;
--
-- Rollback outstanding transaction for the current connection.
PROCEDURE TDF_ROLLBACK
        (SQLCODE);
        ROLLBACK;
```

This problem has been corrected in Oracle Rdb V7.0.

3.2.59 COMMIT with List Cursor Processing No Longer Bugchecks

In previous versions, a COMMIT statement sometimes failed with a bugcheck dump, such as one shown in the following example:

```
***** Exception at 00199B26 : RDMS$$BLOB_CLOSE + 00000006
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=00000078, PC=00199B26, PSL=03C00000
```

This was the result of the commit code calling SQL$CLOSE_CURSORS to close open cursors. It occurred under the following conditions:

- A read LIST cursor was opened on a row but was never explicitly closed.
- A row was deleted (usually by the DELETE ... WHERE CURRENT OF clause).
- An insert LIST cursor was opened and closed.
- A commit was executed.

The problem occurred because all LIST cursors are implicitly closed when the source row is deleted. Thus, the SQL$CLOSE_CURSORS routine was attempting to close an already closed cursor. The workaround to this problem was to explicitly close all LIST cursors before deleting the source row.

This problem has been corrected in Oracle Rdb V7.0.

3.2.60 COMMIT and ROLLBACK Are Now Ignored If There Is No Transaction

In previous versions, an exception was raised when the COMMIT or ROLLBACK statements executed in a multistatement procedure or a stored procedure if there was no active transaction. The ANSI/ISO standard for SQL states that, in these cases, the COMMIT and ROLLBACK statements should be ignored.

Oracle Rdb V7.0 ignores the COMMIT or ROLLBACK statements when no transaction is active.

Because checks for an active transaction are no longer needed, you can simplify your application programming code to make it more readable. The next two code examples show how you might reduce the complexity of your application programs. For example, your programs might include the following code that checks for an active transaction:

```
BEGIN
    DECLARE :ta INTEGER;
    -- Check if a transaction is active
    GET DIAGNOSTICS :ta = TRANSACTION_ACTIVE;
    IF :ta = 1 THEN
        -- Clear previous transaction.
        COMMIT;
    END IF;
    -- Now do the work of the procedure.
    SET TRANSACTION READ ONLY;
    -- Some action
    COMMIT;
END;
```

To simplify the application code, remove the GET DIAGNOSTICS statement and the IF statement as shown in the following example:

```
BEGIN
    -- Clear previous transaction.
    COMMIT;
    -- Now do the work of the procedure.
    SET TRANSACTION READ ONLY;
    -- Some action
    COMMIT;
END;
```

3.3 Oracle RMU Errors Fixed

This section describes problems that have been fixed in the Oracle RMU interface.

3.3.1 RMU Convert Command Works Properly with Fixed-Size AIJ Files

In V6.1, using the RMU Convert command on databases with fixed-size (circular) AIJ journaling resulted in AIJ files being inaccessible at AIJ switch-over time. This resulted in an %RDMS-F-AIJTERMINATE error and the forced exit of the image to protect database integrity.

The workaround was to drop and recreate the .aij files under the converted Oracle Rdb database before resuming database activity.

This problem has been corrected in V7.0.

3.3.2 RMU Open Command and Global_Buffers Qualifier Now Works Correctly

OpenVMS VAX OpenVMS Alpha In previous versions, if the RMU Open command specified the Global_Buffers qualifier but global buffers were not enabled for the database, Oracle Rdb opened the database with no global buffers but did not notify the user that global buffers were not being used.

This problem has been corrected in V7.0. When the RMU Open command specifies the Global_Buffers qualifier, but global buffers are not enabled for the database, Oracle Rdb does not open the database and, instead, returns an error message to the user. ♦

3.3.3 RMU Restore Just_Pages No Longer Leaves Pages with Bad Logical Area

Under certain conditions, the RMU Restore Just_Pages command could leave uniform format pages in the database with a zero logical area number. This problem sometimes resulted in bugchecks in DIOFREE$UNMARK_AIJ_RECORDS when records were inserted into the page after it had been restored.

The following example shows how this could occur:

```
$ SQL
SQL> CREATE DATABASE FILENAME 'TEST'                    -- create a database
cont>    RESERVE 1 JOURNAL
cont>    CREATE STORAGE AREA RDB$SYSTEM FILENAME 'RDB$SYSTEM.RDA'
cont>    PAGE FORMAT IS UNIFORM
cont>    PAGE SIZE IS 4 BLOCKS ALLOCATION IS 200 PAGES
cont>    SNAPSHOT FILENAME 'RDB$SYSTEM' SNAPSHOT ALLOCATION IS 10 PAGES;
SQL> DISCONNECT ALL;
SQL> ALTER DATABASE FILENAME TEST
cont>    JOURNAL IS ENABLED ADD JOURNAL AIJ1 FILENAME AIJ1;
SQL> EXIT
$ RMU/BACKUP TEST BCK                                  ! backup the database
$ SQL
SQL> ATTACH 'FILENAME TEST';
SQL> CREATE TABLE T (T CHAR(1));                       -- create a new table
SQL> COMMIT;
SQL> INSERT INTO T VALUE ('1') RETURNING DBKEY;        -- insert a row
                  DBKEY
               47:186:0
1 row inserted
SQL> COMMIT;
SQL> EXIT;
$ RMU/RESTORE/AREA BCK RDB$SYSTEM/JUST_PAGES=186       ! restore the page
$ RMU/VERIFY/ALL TEST                                  ! verify the database
%RMU-W-BADPTLARE, invalid larea for uniform format data page 186
                  SPAM larea_dbid: 47, page larea_dbid: 0
```

This problem has been corrected in V7.0.

3.3.4 Incremental Restore Now Marks Page Ranges as Changed

In previous versions, when an incremental backup file was restored, it did not mark the page ranges for the restored pages as changed. This occurred only when you used the fast incremental backup option (the default). Thus, if another incremental backup was then done on the restored database, Oracle RMU did not backup the pages restored in the previous incremental restore operation. Consider the following sequence of events:

1. You perform a full backup of the database.
2. Users update the database.
3. You perform an incremental backup operation using the fast incremental backup option (the default).
4. The database is deleted.
5. You restore the full backup file.
6. You restore the incremental backup file.
7. Users update the database.
8. You perform an incremental backup operation using the fast incremental backup option.

In the previous sequence, the last incremental backup operation backed up the database updates made in the previous step, but it did not back up the changes restored in the previous incremental restore operation. If the full backup file was restored again, and the last incremental backup file was restored, the database updates applied by the incremental restore operation done before the last incremental backup were lost.

This problem has been corrected in V7.0.

3.3.5 No Operator Request Issued When Loader Becomes Empty

OpenVMS VAX OpenVMS Alpha

On OpenVMS V6.1 systems and later (or possibly older OpenVMS systems that have had MOUNT patches applied), if the tape loader became empty, Oracle RMU waited for hours before issuing an operator request asking for another tape. For example, when using the RMU Backup command to back up to a device that has a loader, such as a TF867, after all of the tapes in the stacker were used, Oracle RMU waited for more than three hours before it sent a REPLY message to the operator asking for the next tape to be mounted.

The problem was due to changes made to the $MOUNT system service. Prior to OpenVMS V6.1, the $MOUNT system service always returned an error immediately if there was no tape in the tape drive. At the request of OpenVMS customers, the $MOUNT system service was changed to behave differently if the tape drive had a loader. In that situation, the $MOUNT system service continuously polls the drive checking to see if a tape has been loaded. The $MOUNT system service displays the MOUNT-I-WAITDEVRDY message when it begins the polling. If, after five minutes, no tape is loaded, the $MOUNT system service returns an error.

In the past, Oracle RMU always attempted to poll tape drives that had loaders. That is, it called the $MOUNT system service and if the $MOUNT system service returned an error, Oracle RMU waited five seconds and tried again. Depending on the tape drive, Oracle RMU could iterate through that loop 40 or more times. When the $MOUNT system service immediately returned an error, Oracle RMU sometimes waited for close to three minutes before sending an OPERATOR message requesting the next tape. However, with the new changes to the $MOUNT system service, Oracle RMU waited 40 times (*) 5 minutes, or more than three hours, before displaying a message to the operator if there was no tape in the drive.

In V7.0, Oracle RMU has been modified so that it works with the new or old $MOUNT system service behavior. Instead of using a loop count with the $MOUNT system service, Oracle RMU checks elapsed time. If no tape is mounted in approximately three minutes, Oracle RMU displays a message to the operator. ♦

3.3.6 Failed AIJ Backup No Longer Causes Recovery Problems

In previous versions, after an AIJ backup failure, subsequent attempts to recover a database using the failed AIJ backup file did not work correctly under certain circumstances and resulted in portions of transactions not being applied to the database properly.

The following describes one scenario where the AIJ backup failed with the exception AIJJRNBSY and subsequent attempts to recover that backup file did not work properly.

As a consequence of a disk failure (stripe set), the customer had to restore and recover the database. The database restore operation worked correctly, as did the AIJ recovery operation, up to journal #10. Then, instead of continuing to journal #11, the recovery operation asked again for journal #9 and #10, ignoring all the transactions on those journals. The recovery process started automatic AIJ recovery from journal #11 (AIJ_2) which was empty, then asked for journal #12 and ignored all transactions on it and finished the recovery successfully.

This problem resulted because the journal #10 AIJ backup operation failed because journal #11 was busy and backup of journal #11 did not finish (aborted).

This problem has been corrected in V7.0.

The AIJ recovery problem resulted from the AIJ backup operation failure. The AIJ backup operation has been fixed to identify the successful backup of each journal in a multijournal backup operation. This allows the subsequent AIJ recover operation to distinguish between errors trying to start the AIJ backup operation (such as AIJJRNBSY) and errors that occur while actually backing up an AIJ journal (such as disk error).

The previous change required a change in the AIJ recovery operation to allow the processing of appended AIJ journals. This change was available in Oracle Rdb V6.1 and higher.

Following detection of the failed AIJ backup, the AIJ recover operation allowed the recovery sequence number to be incremented incorrectly. This problem has also been corrected.

During testing of the previous two problems, another AIJ recovery problem was identified that resulted in the AIJ backup sequence number being reset to a lower value than that created by the database restore operation. This problem has also been corrected.

An AIJ recover operation for which existing modified AIJ journals exist, such as might occur following a database restore operation, causes the AIJ recovery operation to simulate an AIJ backup operation by creating a new version of the existing AIJ journal. The new journal is created so that the data in the existing AIJ journal is not lost. This allows repeated database restore operations to work properly.

Note

The result of this change is that the AIJ recovery operation can create new AIJ journals following a database restore operation. Ensure that adequate disk space exists, or the AIJ recovery operation will terminate.

Consider the following scenario. A database with five AIJ journals fails due to disk failure. At the time of the failure, the AIJ journal state is the following:

```
Node: BONZ         Oracle Rdb V6.1-00 Performance Monitor   20-JAN-1996 06:17:18
Rate: 1.00 Second             AIJ Information             Elapsed: 00:00:06.79
Page: 1 of 1     KODA$:[R_ANDERS.WORK]MF_PERSONNEL.RDB;2      Mode: Online
--------------------------------------------------------------------------------
Journaling: Enabled   Shutdown:   60  Notify: Enabled   State: Accessible
ALS: Manual     ABS: Disabled  ACE: Disabled  FC: Enabled   CTJ: Disabled

After-Image.Journal.Name....... SeqNum AIJsize CurrEOF Status. State.......
RICK1                           Unused     512   Empty Latent  Accessible
RICK2                           Unused     512   Empty Latent  Accessible
RICK3                           Unused     512   Empty Latent  Accessible
RICK4                                4 *BACKUP NEEDED* Written Accessible
RICK5                                5     512 Unknown Current Accessible
Available AIJ slot 1
Available AIJ slot 2
Available AIJ slot 3
Available AIJ slot 4
Available AIJ slot 5
Available AIJ slot 6
```

In the previous screen, journal #5 is current while journal #4 needs backup.

The following screen shows the AIJ state after the database restore operation:

```
Node: BONZAI         Oracle Rdb V6.1-00 Performance Monitor   20-JAN-1996 06:21:37
Rate: 1.00 Second              AIJ Information           Elapsed: 00:00:07.92
Page: 1 of 1     KODA$:[R_ANDERS.WORK]MF_PERSONNEL.RDB;2      Mode: Online
--------------------------------------------------------------------------------
Journaling: Enabled   Shutdown:   60  Notify: Enabled   State: Accessible
ALS: Manual     ABS: Disabled  ACE: Disabled  FC: Enabled   CTJ: Disabled

After-Image.Journal.Name....... SeqNum AIJsize CurrEOF Status. State.......
RICK1                                6     512 Unknown Current Accessible
RICK2                           Unused     512   Empty Latent  Accessible
RICK3                           Unused     512   Empty Latent  Accessible
RICK4                                4 *BACKUP NEEDED* Written Accessible
RICK5                                5 *BACKUP NEEDED* Written Accessible
Available AIJ slot 1
Available AIJ slot 2
Available AIJ slot 3
Available AIJ slot 4
Available AIJ slot 5
Available AIJ slot 6

--------------------------------------------------------------------------------
```

Note that AIJ journaling was enabled by the database restore operation, which results in journal #6 being activated and becoming current. However, recovery of the AIJ journals begins with AIJ sequence #0, as described by the RMU Dump Header command.

The following example shows the normal results of recovering the first AIJ backup file:

```
$ RMU/RECOVER/LOG backup1.aij
%RMU-I-LOGRECDB, recovering database file KODA1:[R_ANDERS.WORK]MF_PERSONNEL.RDB;2
%RMU-I-LOGOPNAIJ, opened journal file KODA$:[R_ANDERS.WORK]BACKUP1.AIJ;1
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-LOGRECOVR, 1 transaction committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 1 transaction ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence number
    needed will be 1
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-LOGSUMMARY, total 1 transaction committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
%RMU-I-LOGSUMMARY, total 1 transaction ignored
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence number
    needed will be 1
```

The following example shows the results of recovering the second AIJ backup file. This example is important because it demonstrates the new behaviour where a new AIJ journal is created to simulate the backup of AIJ journal #4 and journal #5.

```
$ RMU/RECOVER/LOG backup2.aij
%RMU-I-LOGRECDB, recovering database file KODA1:[R_ANDERS.WORK]MF_PERSONNEL.RDB;2
%RMU-I-LOGOPNAIJ, opened journal file KODA$:[R_ANDERS.WORK]BACKUP2.AIJ;1
```

```
%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed
%RMU-I-AIJONEDONE, AIJ file sequence 2 roll-forward operations completed
%RMU-I-AIJONEDONE, AIJ file sequence 3 roll-forward operations completed
%RMU-I-LOGRECOVR, 6 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence number
    needed will be 4
%RMU-I-AIJAUTOREC, starting automatic after-image journal recovery
%RMU-I-LOGOPNAIJ, opened journal file  KODA1:[R_ANDERS.WORK]RICK4.AIJ;3
%RMU-I-AIJONEDONE, AIJ file sequence 4 roll-forward operations completed
%RMU-W-AIJDEVDIR, AIJ filename "RICK4.AIJ" does not include device and
    directory
%RMU-I-LOGCREAIJ, created after-image journal file
KODA1:[R_ANDERS.WORK]RICK4.AIJ;4
%RMU-I-LOGMODSTR,     created after-image journal "RICK4"
%RMU-I-LOGRECOVR, 3 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence number
    needed will be 5
%RMU-I-LOGOPNAIJ, opened journal file KODA1:[R_ANDERS.WORK]RICK5.AIJ;2
%RMU-I-AIJONEDONE, AIJ file sequence 5 roll-forward operations completed
%RMU-W-AIJDEVDIR, AIJ filename "RICK5.AIJ" does not include device and
    directory
%RMU-I-LOGCREAIJ, created after-image journal file
    KODA1:[R_ANDERS.WORK]RICK5.AIJ;3
%RMU-I-LOGMODSTR, created after-image journal "RICK5"
%RMU-I-LOGRECOVR, 0 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJNOACTIVE, there are no active transactions
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence number
    needed will be 5
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-LOGSUMMARY, total 9 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
%RMU-I-LOGSUMMARY, total 0 transactions ignored
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence number
    needed will be 5
```

The following example from the Performance Monitor shows the AIJ state after the database recovery operation:

```
Node: BONZAI          Oracle Rdb V6.1-00 Performance Monitor   20-JAN-1996 06:32:52
Rate: 1.00 Second               AIJ Information            Elapsed: 00:00:06.98
Page: 1 of 1     KODA$:[R_ANDERS.WORK]MF_PERSONNEL.RDB;2      Mode: Online
--------------------------------------------------------------------------------
Journaling: Enabled   Shutdown:   60  Notify: Enabled   State: Accessible
ALS: Manual     ABS: Disabled  ACE: Disabled  FC: Enabled   CTJ: Disabled

After-Image.Journal.Name....... SeqNum AIJsize CurrEOF Status. State.......
RICK1                                6     512 Unknown Current Accessible
RICK2                           Unused     512   Empty Latent  Accessible
RICK3                           Unused     512   Empty Latent  Accessible
RICK4                           Unused     512   Empty Latent  Accessible
RICK5                           Unused     512   Empty Latent  Accessible
Available AIJ slot 1
Available AIJ slot 2
Available AIJ slot 3
Available AIJ slot 4
Available AIJ slot 5
Available AIJ slot 6
```

```
--------------------------------------------------------------------------------
```

As you can see, AIJ journals sequence #4 and sequence #5 have been backed up.

However, in the directory there are two RICK4 and RICK5 AIJ journals. RICK4.AIJ;3 and RICK5.AIJ;2 are the original AIJ journals known to the database backup operation. RICK4.AIJ;4 and RICK5.AIJ;3 are the new AIJ journals created by the AIJ recover operation to simulate the backup of the original AIJ journals in a nondestructive manner.

3.3.7 Performance Monitor Uses RDMS$BIND_STATS_DISABLED Correctly

In previous versions, if the RDMS$BIND_STATS_DISABLED logical name or the RDB_BIND_STATS_DISABLED configuration parameter was defined, the Performance Monitor did not allow the user to monitor the database.

This problem has been corrected in V7.0.

3.3.8 Fixed-Size AIJ Backup to Tape No Longer Ignores Active Checkpoints

In previous versions, when using the Format=New_Tape qualifier to back up fixed-size .aij files, regardless of whether the backup file was a magnetic tape or a disk device, the AIJ backup utility did *not* wait for active checkpoints to migrate from the .aij file being backed up.

This problem could result in the database being nonrecoverable if any of the processes with the active checkpoint failed subsequent to the AIJ backup operation. The database recovery (DBR) process would be unable to redo the failed processes' committed transactions.

The following example shows a user with an active checkpoint on the .aij file that has a sequence number of 15. Then, the AIJ backup command successfully backs up this .aij file, even though it is not supposed to do so.

```
$ RMU/DUMP/USER DATABASE.RDB
Active user with process ID 404003D2
    Stream ID is 1
    Monitor ID is 1
    Transaction ID is 361
    Recovery journal filename is
    "EBS$PRODUCTION_DATA_ROOT1:[RDM$RUJ]EXC_DB$0099899462E20C2B.RUJ;1"
    Read/write transaction in progress
    Last AIJ checkpoint 15:3     <=================
    Transaction sequence number is 170611

$ RMU /BACKUP /AFTER /QUIET /LOG /LABEL=(PRDB01) -
    /FORMAT=NEW_TAPE DATABASE.RDB $1$MUA0:P01EXC_DB951031
%RMU-W-NAMTRUNC, File name truncated, $1$MUA0:[000000]P01EXC_DB951031.AIJ_RBF;
%RMU-I-AIJBCKBEG, beginning after-image journal backup operation
%RMU-I-AIJBCKSEQ, backing up after-image journal sequence number 15
%RMU-I-LOGBCKAIJ, backing up after-image journal EXC_AIJ_03 at 02:13:57.55
%RMU-I-AIJBCKEND, after-image journal backup operation completed successfully
%RMU-I-LOGAIJJRN, backed up 1 after-image journal at 02:15:16.57
%RMU-I-LOGAIJBLK, backed up 113997 after-image journal blocks at 02:15:16.57
%RMU-W-NAMTRUNC, File name truncated, $1$MUA0:[]P01EXC_DB951031.AIJ_RBF;
```

The workaround was to back up the .aij files to disk, using the Format=Old_Rms qualifier, then copy the disk backup file to magnetic tape.

This problem has been corrected in V7.0. The AIJ backup utility now properly waits for checkpoint migration for all types of backup media operations.

3.3.9 DBR No Longer Bugchecks During Extensible AIJ Backup

In previous versions, under extremely rare circumstances, the database recovery (DBR) process could have bugchecked attempting to redo a transaction for a failed process.

This problem only occurred during an AIJ backup of an extensible .aij file when the fast commit feature was enabled.

This problem has been corrected in V7.0. The DBR process correctly redoes the transaction of the failed process.

3.3.10 RMU Backup Command No Longer Deadlocks During Extensible AIJ Backup

In previous versions, a deadlock situation could have occurred while backing up an extensible .aij file.

The problem occurred when the Fast Commit option was enabled and at least one active checkpoint operation was found. The problem occurred only when the AIJ data needed to be moved from the active AIJ file to the new AIJ file, such that an AIJ extension operation occurred.

The workaround was to increase the size of the AIJ journal initial allocation.

This problem has been corrected in V7.0.

3.3.11 Quiet-Point AIJ Backups Spanning Transactions Can Now Be Applied

In previous versions, in certain situations, an .aij file created using a quiet-point backup could not be rolled forward by itself because it spanned transactions. The journal had to be applied in conjunction with some previous .aij file. This problem involved the RMU Recover, RMU Backup After_Journal, RMU Copy, RMU Move, and RMU Set After_Journal commands.

This problem has been corrected in V7.0.

3.3.12 Time Reduced for AIJ Journal Creation and Extension

In previous versions, the time needed to create an .aij file or to extend a fixed-size .aij file was excessive. Preliminary analysis showed that .aij files were created at an approximate average rate of 500-650 blocks per second. This problem involved the RMU Set After_Journal, the RMU Backup After_Journal commands, and the SQL and RDO interfaces.

The following table shows performance measurements obtained when a 96,000 block .aij file was created during a relatively light usage period:

Oracle Rdb V6.1	Test #1	Test #2	Test #3	Test #4
#Buffered I/O	79	79	79	80
#Direct I/O	1076	1076	1059	1020
Elapsed CPU Time	1.46	1.43	1.70	1.51
Elapsed Wall Time	1:28.61	2:03.66	2:53.66	2:47.03
CPU Utilization	1%	1%	0%	0%
Blocks Per Second	1090	780	554	574

The .aij file creation and extension algorithm has been optimized to use overlapping initialization and I/O operations. The result is a substantial decrease in the overall time required to create or to extend .aij files

Using the same data, the following table shows the new performance information. Performance results depend on a variety of factors—your actual performance may vary from the information presented here.

Oracle Rdb V7.0	Test #1	Test #2	Test #3	Test #4
#Buffered I/O	80	80	80	80
#Direct I/O	1023	864	861	864
Elapsed CPU Time	2.31	2.20	2.37	2.32
Elapsed Wall Time	56.70	54.92	54.46	54.19
CPU Utilization	4%	4%	4%	4%
Blocks Per Second	1714	1777	1777	1777

Further analysis reveals that, depending on the disk device used for testing, 1777 blocks per second is the highest achievable performance possible. Obviously, performance increase scales according to the performance of the disk device used.

3.3.13 Determining the True Size of Current .aij File

OpenVMS VAX OpenVMS Alpha

After upgrading to Oracle Rdb V6.0-5, a user could no longer determine the size of the current .aij file. The user had a process running on Oracle Rdb V4.x that could watch the number of used blocks of an .aij file and, when a threshold was crossed, the process started an automatic .aij backup (which then reset the number of used blocks downward). After upgrading to Oracle Rdb V6.0-5, the number of allocation blocks and the number of used blocks are equal, therefore the user's process tried to back up the .aij file every time it woke up. This problem involved the RMU Show After_Journal Backup_Context and RMU Backup After_Journal commands.

The workaround was to parse the output from the RMU Dump Header command. The RMU Dump Header command displayed information about the size of each .aij file, including its percentage of fullness.

The problem has been corrected in V7.0. Two new process global symbols have been added to the RMU Show After_Journal command with the Backup_Context qualifier and the RMU Backup After_Journal command. They provide a reliable method to determine size of current .aij file, as follows:

- RDM$AIJ_ENDOFFILE—identifies the end of file block number for the current .aij file
- RDM$AIJ_FULLNESS —identifies the percentage of fullness of the current .aij file

These new symbols work for both extensible and fixed-size .aij files.

The following example shows the command that initializes the backup context and defines the process global AIJ symbols:

```
$ RMU/SHOW AFTER_JOURNAL/BACKUP_CONTEXT/OUT=NL: MF_PERSONNEL
```

The following example shows the output from the DCL SHOW SYMBOL command:

```
$ SHOW SYMBOL RDM$AIJ*
  RDM$AIJ_COUNT == "5"
  RDM$AIJ_CURRENT_SEQNO == "10"
  RDM$AIJ_ENDOFFILE == "329"   <-- NEW!
  RDM$AIJ_FULLNESS == "64"     <-- NEW!
  RDM$AIJ_LAST_SEQNO == "9"
  RDM$AIJ_NEXT_SEQNO == "6"
```

In the previous example, the .aij file whose sequence number is "10" (RDM$AIJ_CURRENT_SEQNO) is currently 64% full (RDM$AIJ_FULLNESS) at 329 blocks (RDM$AIJ_ENDOFFILE).

Remember that the process global symbols are created as strings, and must be converted into numeric symbols if they are to be compared with computed symbols. ♦

3.3.14 AIJ Rollforward Can Start from Quiet-Point AIJ Backup or Quiet-Point Database Backup

In previous versions, the rollforward of an .aij file always started with an AIJ backup quiet-point, even if the database was restored using a quiet-point database backup.

The following figure demonstrates the problem more clearly:

```
  AIJ0              AIJ1                                  AIJ2
   |                 |                                     |
  Time--------------[1]------------------[2]--------------[3]-----[4][5]----->
            |        |           |        |       |        |
            |        |           |        |       |        |
            |    Backup AIJ      |   Database     |    Backup AIJ
            |    No Quiet-Point  |   Full Backup  |    No quiet-Point
            |                    |                |
            |<------------------>|                |<---------------
       Start Long Update     End Update      Start Long Update
       Transaction           Transaction     Transaction
```

The following explains the steps Oracle Rdb used:

1. Performed a no-quiet-point AIJ backup (using the RMU Backup After_Journal command with the Noquiet_Point qualifier) for AIJ0 (current). At that moment, transactions spanned AIJ0 and AIJ1.
2. Performed a full quiet-point database backup.
3. Performed a no-quiet-point AIJ backup for AIJ1 (current). At that moment, transactions spanned AIJ1 and AIJ2.
4. The database became unavailable due to catastrophic node failure.
5. Immediately started a restore from a full database backup file [4] and subsequent AIJ recovery.

The following AIJ journal sequence information was extracted using the RMU Dump After command:

```
 - AIJ0
   AIJ Sequence Number is 0
   Open mode is Initial

 - AIJ1
   AIJ Sequence Number is 1
   Transaction from AIJ sequence number 0 may span into this journal
   Open mode is Continuation
```

At the time of the full database backup (Step 2), the current .aij file was AIJ1, so both AIJ1 and AIJ2 were needed for the recovery (Step 5). However, the RMU Recover command required AIJ0 as a first .aij file, even though the database was restored using the quiet-point database backup file.

The workaround was to start the AIJ rollforward operation using the latest quiet-point AIJ backup. It was recommended that a quiet-point AIJ backup operation *always* preceed a quiet-point database backup operation. In addition, if you were going to perform a quiet-point database backup, it was better to perform a quiet-point AIJ backup operation followed by a no-quiet-point database backup operation. This resulted in the same number of quiet-points and simplified the AIJ rollforward requirements.

This problem has been corrected in V7.0.

The AIJ rollforward operation can start from either a quiet-point AIJ backup or a no-quiet-point AIJ backup:

- If a no-quiet-point database backup file is restored, the AIJ rollforward operation must start from the preceding quiet-point AIJ backup file.
- If a quiet-point database backup file is restored, the AIJ rollforward operation can start from the .aij file that was active at the time of the database backup operation, even if it was not created with a quiet-point AIJ backup operation.

You can determine if the the backup file was restored from a quiet-point or no-quiet-point backup operation by using the RMU Dump Header command.

The RMU Dump Header command displays the following message when a no-quiet-point database backup file has been restored, indicating that you must start the AIJ rollforward from a quiet-point AIJ backup file:

```
AIJ roll-forward is no-quiet-point enabled
```

The RMU Dump Header command displays the following message when a quiet-point database backup file has been restored, allowing you to start the AIJ rollforward from a quiet-point or no-quiet-point AIJ backup file:

```
AIJ roll-forward is quiet-point enabled
```

3.3.15 RMU Set Audit Stop Command Now Stops Auditing of RMU Commands

OpenVMS VAX OpenVMS Alpha

In previous versions, if auditing of RMU commands was enabled for a database and auditing was started, the RMU Set Audit Stop command did not stop auditing RMU commands. The command did stop other types of auditing correctly.

The following is an example of unwanted alarms:

```
$ REPLY/ENABLE=SECURITY
$ RMU/SET AUDIT/EVERY/FLUSH/ENABLE=RMU mf_personnel
$ RMU/SET AUDIT/START mf_personnel
$!
$! The following should have stopped any alarms:
$ RMU/SET AUDIT/STOP  mf_personnel
```

A workaround was to use the RMU Set Audit command with the Disable qualifier.

This problem has been corrected in V7.0. ♦

3.3.16 Bugcheck at RDMS$$KOD_ISCAN_START_SCAN + AC

OpenVMS Alpha

In previous versions, the RMU Verify Constraints command sometimes failed on OpenVMS Alpha with a bugcheck at RDMS$$KOD_ISCAN_START_SCAN + AC because of some particular constraints and indexes.

The following example shows the problem:

```
SQL> CREATE DATABASE FILENAME TST
cont>  CREATE TABLE t (
cont>     F1 INTEGER,
cont>     F2 INTEGER,
cont>     CONSTRAINT C_CHECK1
cont>         CHECK(((T.F1 <> 0)
cont>             AND (NOT (T.F1 IS NULL))))
cont>         DEFERRABLE);
SQL> CREATE INDEX I_f1 on t (f1);
SQL> COMMIT;
```

```
$ RMU/VERIFY/CONSTRAINT TST
%RDMS-I-BUGCHKDMP, generating bugcheck dump file disk:[dir]RDSBUGCHK.DMP;
%RMU-W-CNSTVERER, verification of constraints is incomplete due to errors.
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=500196A0, PC
=00000000, PS=00000000
```

A workaround was to drop the index before verifying the constraints.

This problem has been corrected in V7.0. ♦

3.3.17 Oracle RMU for Digital UNIX Now Translates Environment Variables in .sqlrc Configuration File

Digital UNIX

In V6.1, Oracle RMU for Digital UNIX did not translate environment variables in the .sqlrc file for database names given on the command line. This was inconsistent with SQL behavior and was inconvenient for users.

This problem has been corrected for V7.0 on Digital UNIX. ♦

3.3.18 RMU Show Statistics with Input Qualifier No Longer Fails

OpenVMS VAX OpenVMS Alpha

In previous versions, when the RMU Show Statistics command with the Input=file qualifier was used to display a binary statistics file collected on another system, RMU sometimes failed while attempting to find database files to which the binary collection file referred.

The following example shows how this problem occurred:

```
On system S1:
```

```
$ RMU/SHOW STAT/OUTPUT=MAIN.STATS/OPTIONS=(BASE,AREA) PERSONNEL
```

```
On system S2:
```

```
$ COPY S1::MAIN.STATS []
$ RMU/SHOW STAT/INP=MAIN.STATS
%RMS-F-DNF, Directory not found
-SYSTEM-W-NOSUCHFILE, No such file
%RMU-F-FATALOSI, Fatal error from the operating system interface.
```

A workaround was to display the binary statistics file from the same system where it was collected.

This problem has been corrected in V7.0. Oracle RMU now attempts to find the database file on the system, and if it does not, Oracle RMU continues rather than aborting. ♦

3.3.19 Performance Monitor Starts Up More Quickly

In previous versions, the time required to display the first Performance Monitor screen increased as the number of database storage areas increased.

This problem has been corrected in V7.0. The Performance Monitor has been enhanced to perform as little up-front initialization as possible. Now, the screen initialization algorithms are performed when the screen is displayed for the first time. As a result, the Performance Monitor never initializes the screens that are not actually displayed.

Note

Selecting Options on the horizontal menu to write all screens causes each screen's initialization to be performed.

3.3.20 Oracle RMU Now Clears Snapshot, SPAM Pages from CPT During RMU Repair Command

Oracle Rdb inserts an entry in the corrupt page table (CPT) for any corrupt page it encounters. The RMU Repair command can be used to repair these corrupt pages, but, in previous versions, the corresponding entries in the CPT were not removed for snapshot, SPAM, or ABM pages. In addition, the RMU Set Corrupt_Pages command did not allow manipulation of snapshot pages, preventing snapshot page entries from being manually removed from the CPT.

The RMU Repair command has been corrected to clear the entries in the CPT for pages it has repaired. The RMU Set Corrupt_Pages command has been modified so that snapshot pages can be removed manually from the CPT.

3.3.21 Corrupt Pages in Snapshot Areas Can Now Be Removed

Beginning with V6.0, Oracle Rdb maintained a corrupt page table (CPT) to record pages that had checksum errors on them. Entries were removed from the CPT when the page was no longer corrupt, usually a result of an RMU Restore operation. However, entries for corrupt pages in snapshot areas could not be removed. This caused bugchecks for any process that tried to read the pages marked as corrupt.

This problem has been corrected in V7.0. You can now remove entries for snapshot pages from the CPT by using the Initialize=Snapshot qualifier of the RMU Repair command. Oracle Rdb removes any entries in the CPT for snapshot areas that are initialized by this command.

3.3.22 RMU Monitor Logging No Longer Disables Over Time

OpenVMS VAX OpenVMS Alpha

In previous versions, monitor logging disabled itself over time for no apparent reason when you used the RMU Monitor command, even though there was ample space on the log device. Occasionally, restarting the log worked, but only for a short time. This problem usually occurred when the monitor was very busy.

This problem has been corrected in V7.0. ♦

3.3.23 RMU Analyze Cardinality Update Command Correctly Updates Cardinality

OpenVMS Alpha ☰ In previous versions, the RMU Analyze Cardinality Update command did not always update cardinality correctly. This command correctly updated the cardinality only when a table that had compression enabled or a hash index was included in the list of tables and indexes to analyze. If the list included at least one such table or index, the cardinality for all tables and indexes was updated correctly.

The workaround was to include a table that had compression enabled or a hash index in the list of objects to analyze when you updated cardinality.

This problem has been corrected in V7.0. ♦

3.3.24 Changes to Header in RMU Analyze Output

Minor changes have been made to the output from the RMU Analyze command. Now, RMU Analyze prints the current date and time in the header for the output. This allows you to know how recent or stale the data is in an output file. Also, when generating area reports, RMU now includes the name of the database in the report header. The name of the database is already displayed for index reports and placement reports. Note that the examples in the documentation do not show these new features because they are late-breaking product enhancements.

The following example shows the start of the output for an area report generated by RMU Analyze.

```
$ RMU /ANALYZE MF_PERSONNEL

Areas for database - USER:[DB]MF_PERSONNEL.RDB;1
Created 16-AUG-1996 01:18:06.70

----------------------------------------------------------------------------

Storage analysis for storage area: RDB$SYSTEM - file: US:[DB]MF_PERS_DEFAULT.RDA
   .
   .
   .
```

3.3.25 RMU Unload Command No Longer Pads VARCHAR Fields When Producing Delimited Text Files

In previous versions, VARCHAR columns expanded to their maximum size when the RMU Unload command produced a delimited text output file.

This problem has been corrected in V7.0; VARCHAR columns are no longer padded when unloaded into a delimited text unload file.

For example, if "ABC" is stored in a VARCHAR(10) column, previous versions of Oracle Rdb produced the following formatting for the RMU Unload command:

```
"ABC       ".
```

Now, the RMU Unload command shows the column without the padding:

```
"ABC".
```

There was no workaround for this problem. Users who need to preserve the old behavior within V7.0 must unload data through a view of the unloaded table in which all VARCHAR fields are cast as CHAR fields.

3.3.26 RMU Checkpoint Command Now Allows No Wait for Completion

In previous versions, the RMU Checkpoint command always waited for the total system checkpoint completion. Often, it was desirable to not perform the checkpoint synchronously, but *try* to see if the checkpoint operation was possible.

This problem has been corrected in V7.0. The RMU Checkpoint command now accepts a [No]Wait qualifier, which indicates whether or not the utility is to wait for checkpoint completion. The default qualifier is Wait.

3.3.27 DBR No Longer Bugchecks Trying to Fetch Inconsistent Pages

In previous versions, when a user atempted to recover an inconsistent page with the RMU Recover command, the database recovery (DBR) process crashed attempting to fetch the inconsistent page. The RMU Recover command attempted an attach that triggered the DBR process, which bugchecked trying to access the inconsistent page. The monitor shut down the database and the RMU Recover command was forced out of the database.

The following shows an example of this problem:

```
$ SQL$
SQL> ALTER DATABASE FILE mf_personnel
cont>    JOURNAL FILE jour1.aij
cont>    JOURNAL ALLOCATION IS 20000 BLOCKS;
SQL> EXIT
$ RMU/BACKUP mf_personnel mf_personnel
$ sql$
SQL> ATTACH 'FILE mf_personnel';
SQL> -- get the area and page numbers that contain relevant records
SQL> SELECT DBKEY FROM EMPLOYEES WHERE EMPLOYEE_ID > '00400';
SQL> COMMIT;
SQL> UPDATE EMPLOYEES SET SEX='M' WHERE EMPLOYEE_ID > '00400';
SQL> COMMIT;
SQL> DELETE FROM EMPLOYEES WHERE EMPLOYEE_ID > '00400';
```

At this point, suppose the monitor on the current node was killed (as in a node crash). If the database was not in use on any other node, the recovery of the user takes place only when another attach was requested.

Suppose one of the pages affected by the deletion operation was page 371 of storage area EMPIDS_OVER. Suppose you made the page corrupt and then restored it so that it was marked inconsistent:

```
$ RMU/ALTER mf_personnel
RdbALTER>  area EMPIDS_OVER
RdbALTER> page 371
RdbALTER> dep checksum=12345678
RdbALTER> commit
RdbALTER> exit
$
$ RMU/RESTORE/NORECOVER/AREA mf_personnel.rbf EMPIDS_OVER/JUST_PAGES=371
```

Next, suppose you tried to recover the page (to make it consistent). The attach caused the monitor to start a DBR process to undo the user (who was active when the database was shutdown). However, because that user had touched the same page (page 371 of EMPIDS_OVER) and the page was marked inconsistent, the DBR process crashed when it tried to access the page. Then, the monitor shut down the database. The RMU Recover attach, which waited for the DBR process to complete successfully, was forced out of the database.

```
$ RMU/RECOVER/JUST_PAGES jour1.aij
```

The workaround was to make the user temporarily non-recoverable, by removing the .ruj file name from the root file. Then, the RMU Recover attach did not trigger the DBR process. Once the RMU Recover made the pages and areas consistent, the root file was altered to add the .ruj file name. A subsequent attach request triggered the DBR process and the DBR process recovered the user successfully.

This problem has been corrected in V7.0.

3.3.28 Database Recovery No Longer Runs with DBKEY SCOPE IS ATTACH Properties

In previous versions, the database recovery (DBR) process sometimes bugchecked when trying to insert a record because of a lack of space on the page after a crash.

This occurred when the following conditions were true:

- You attached to the database using the DBKEY SCOPE IS TRANSACTION clause of the SQL ATTACH statement.
- You specified the JOURNAL FAST COMMIT ENABLED clause of the ALTER DATABASE statement.
- The system crashed after the transaction committed and inserted the record into a database page but before the page was written back to disk.

The exception occurred at DBR$DO_C_AIJBUF + 2B0.

The workaround was to avoid the problem by using the DBKEY SCOPE IS ATTACH clause, instead of the DBKEY SCOPE IS TRANSACTION clause. However, once the problem occurred, there was no workaround.

This problem has been corrected in V7.0. The DBR process never considers the dbkey scope to be attach.

3.3.29 RMU Show After_Journal Backup_Context Command Properly Creates and Deletes Process Global Symbols

In V6.1, the process global symbols RDM$AIJ_BACKUP_SEQNO and RDM$AIJ_SEQNO incorrectly remained defined after the following sequence of commands:

```
$ RMU/SHOW AFTER_JOURNAL/BACKUP_CONTEXT DB_WITHOUT_AIJ
$ SHOW SYMBOL RDM$AIJ* => 4 SYMBOLS DEFINED

$ RMU/SHOW AFTER_JOURNAL/BACKUP_CONTEXT DB_WITH_AIJ
$ SHOW SYMBOL RDM$AIJ* => 6 SYMBOLS DEFINED

$ RMU/SHOW AFTER_JOURNAL/BACKUP_CONTEXT DB_WITHOUT_AIJ
$ SHOW SYMBOL RDM$AIJ* => 6 SYMBOLS DEFINED INSTEAD OF 4
```

The problem has been corrected in V7.0. The proper process global symbols are now created and deleted appropriately for the state of the after-image journal (.aij) subsystem.

3.3.30 Now You Can Invoke Many Simultaneous Database Attach and Detach Operations While Using the RMU Show Users Command

Prior to V7.0, a monitor bugcheck error occurred when a monitor performed many simultaneous database attach or detach operations, including database recovery (DBR) processes. The problem occurred more frequently if the operations occurred at the same time you were using the RMU Show Users command.

The following example shows the MONITOR bugcheck problem:

```
...
Saved PC = 00021AC0 : KOD$BUGCHECK_AND_EXIT_HNDLR + 0000002C
ARG# Argument [data...] ----------------------------------------------------
1 7FF09884: 00000000 056EC02C 00000001 03C00000 00016F04 056EC02C 00000003
        ...
Saved PC = 0001A6DC : MONITOR + 0000065C
ARG# Argument [data...] ----------------------------------------------------
1 7FF09884: 00000000 056EC02C 00000001 03C00000 00016F04 056EC02C 00000003
2 7FF0986C: 00000003 03000001 00000002 00000003 00000002 7FF09998 00000004
        ...
***** Exception at 00016F04 : COSI_CHF_SIGNAL + 00000042
%COSI-F-BUGCHECK, internal consistency failure

Handler = 00000000, PSW = 0000, CALLS = 0, STACKOFFS = 0
Saved AP = 7FF0990C, Saved FP = 7FF098E8, PC Opcode = 05
140 bytes of stack data from 7FF0985C to 7FF098E8:
        ...
Saved PC = 80000014 : S0 address
ARG# Argument [data...] ----------------------------------------------------
1 00000001
        ...
Saved PC = 0000ECA1 : COSI_TINCAN_FREE_BUFFER + 00000055
ARG# Argument [data...] ----------------------------------------------------
1 001159DC: 38363037 30323132 00000010 00000001 21207068 0000017C 00133140
2 7FF09958: 00000000 00000000 7FF09974 03040004 00000000 00000001 00000000
        ...
```

The problem has been corrected in V7.0. The user interface has not changed as a result of the correction.

3.3.31 RMU Extract Command with Item=Security Qualifier Now Generates Correct DCL Syntax

OpenVMS VAX OpenVMS Alpha

In previous versions, the table name extracted by the RMU Extract command with the Item=Security qualifier was not processed correctly, and may have contained trailing spaces or other non-printing characters.

The following excerpt of a script generated by the RMU Extract command shows the table named COLLEGES followed by a trailing space:

```
$ RMU/EXTRACT/ITEM=SECURITY MF_PERSONNEL
     .
     .
     .
$ RMU/SET AUDIT/TYPE=AUDIT -
         /ENABLE=DACCESS=COLUMN=(COLLEGES .COLLEGE_CODE,COLLEGES .COLLEGE_NAME) -
         /PRIV=(SELECT,INSERT,UPDATE) -
         RDB$ROOT2:[70]MF_PERSONNEL.RDB;3
```

The workaround was to manually edit the script file generated by the RMU Extract command. This problem has been corrected in V7.0. ♦

3.3.32 RMU Extract Command with the Option Qualifier Processes Keywords Correctly

In V6.1, an error in the Option qualifier on the RMU Extract command caused keywords to be incorrectly translated and resulted in unexpected output. The problem occurred because the Option keyword decoding table incorrectly processed and translated keywords to the wrong internal setting.

This problem has been corrected in V7.0.

Also, note that the Option=Debug qualifier now causes Oracle RMU to output a new option-verification dump that shows the internal settings options, including default options and options that were explicitly specified.

3.3.33 RMU Recover Resolve Can Now Resolve Transactions Originated from Different Systems

OpenVMS VAX OpenVMS Alpha

In previous versions, the RMU Recover Resolve command failed because it could not identify the participant node when it attempted to rollforward an .aij file that contained an unresolved distributed transaction from a system other than the one on which it was originated.

The following example shows the problems that might be caused by this failure:

```
$ RMU/RESTORE/NOCDD/DIR=DISK$:[USER] DB1905.RBF
$ RMU/RECOVER/ROOT=DISK$:[USER]DB.RDB DB0406.AIJ
$ RMU/RECOVER/ROOT=DISK$:[USER]DB.RDB DB.AIJ
```

The second recovery operation ended with the following:

```
%RMU-I-AIJONEDONE, AIJ file sequence 57 roll-forward operations completed
%RMU-I-LOGRECOVR, 35 transactions committed
%RMU-I-LOGRECOVR, 0 transactions rolled back
%RMU-I-LOGRECOVR, 0 transactions ignored
%RMU-I-AIJACTIVE, 1 active transaction not yet committed or aborted
%RMU-I-LOGRECSTAT, transaction with TSN 1385214 is active
%RMU-I-AIJPREPARE, 1 of the active transaction prepared but not yet committed
or aborted
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJNXTSEQ, to continue this AIJ file recovery, the sequence number
needed will be 58
%RMU-F-PARTDTXNERR, error trying to participate in a distributed transaction
-SYSTEM-F-UNREACHABLE, remote node is not currently reachable
$
$ RMU/DUMP/AFTER_JOURNAL/STATE=PREPARED DB.AIJ

1/1              TYPE=O, LENGTH=510, TAD=11-MAY-1995 04:23:37.23, CSM=00
    Database $1$DKB0:[PILBA5.EXP]DB.RDB;1
    Database timestamp is 30-SEP-1993 11:12:11.36
    Facility is "RDMSAIJ ", Version is 601.0
    AIJ Sequence Number is 57
    Last Commit TSN is 1385212
    Synchronization TSN is 0
    Type is Normal (unoptimized)
    Open mode is Initial
    Backup type is Active
    I/O format is Record
    Commit-to-Journal optimization disabled

66/141           TYPE=C, LENGTH=18, TAD=11-MAY-1995 04:30:38.21, CSM=00
    TSN=1385214

2034514353410D9111CE8B49E556C54A       TID: 'JÅVåI.Î...ASCQ4 '
315245474F525E8E11CE3EB345448AED       TM LOG_ID: 'í.DE³>Î..^ROGER1'
00000000000000000000000600000691       RM LOG_ID: '................'
002020202020202030424149 4B0200DD       RM_NAME: '..KIAB0      .'
00000000000600010000011089C0000       RM_NAME: '................'
                     3451435341       NODE NAME: 'ASCQ4'
                     3451435341       PARENT NODE NAME: 'ASCQ4'
```

In this example, the database was left in a corrupt state:

```
SQL> attach 'filename DB';
%SQL-F-ERRATTDEC, Error attaching to database DB
-RDB-F-DB_CORRUPT, database filename is corrupt
-RDMS-F-AREA_CORRUPT, storage area DISK$:[USER]DB.RDB;1 is corrupt
SQL>
```

The workaround was to create a DECnet node on the new system with the same node name used to prepare the transaction.

This problem has been corrected in V7.0. The RMU Recover Resolve command now allows manual resolution of prepared distributed transactions, even on systems not originally involved in the transaction. ♦

3.4 RdbPRE, RDML, and RDO Errors Fixed

This section describes problems that have been fixed in the RdbPRE, RDML, and RDO interfaces.

3.4.1 Alignment of Host Variable Smallint in the RDBPRE Preprocessor

OpenVMS VAX OpenVMS Alpha

In previous versions, with the RDBPRE preprocessor, host variables of the data type smallint (word) were not aligned on word boundaries on OpenVMS and would generate the following error:

```
.INTEGER *2 MESSAGE$1_1_VAR$12
..................^
%FORT-W-MISALIGN, Alignment of variable or array is inconsistent with its data
type
```

The problem occurred when a smallint data type followed a string. The string allocated correctly, but the smallint was not aligned on a word boundary.

This problem is corrected in Oracle Rdb V7.0. ♦

3.4.2 RDO MATCHING Operator Can Now Find Date Matches

OpenVMS Alpha

In previous versions, the RDO MATCHING operator may have failed to find date matches, when applied to certain types of nontext data.

The following example shows a command that would have failed:

```
for s in salary_history with s.salary_start
 matching "1982%%%%00000000" print s.* end_for
```

This problem has been corrected in V7.0. ♦

3.4.3 RDML/Pascal Now Correctly Generates Casting

OpenVMS VAX OpenVMS Alpha

In previous versions, RDML/Pascal incorrectly generated casting for the whole record. For example, for the field PER_REC.BIRTHDAY, RDML generated the following:

```
 RDB$MSG_PORT_1_1.RDB$PORT_FIELD_1
             := PER_REC::RDML$CDDADT_TYPE.BIRTHDAY;
```

Now, it generates the correct casting:

```
 RDB$MSG_PORT_1_1.RDB$PORT_FIELD_1
             := PER_REC.BIRTHDAY ::RDML$CDDADT_TYPE;
```

♦

3.4.4 RDMLVAXC.H Is Now Compatible with C++

OpenVMS VAX OpenVMS Alpha

In previous versions, when you compiled RDML applications with C++, you received errors such as the following:

```
CXX-E_ARGCOUNT RDML$VC_INITIALIZE2 supply 8 arguments when 0 expected.
```

This problem has been corrected in V7.0. ♦

4 Documentation Additions and Changes

This chapter provides descriptions of late-breaking product enhancements and changes and corrections for documentation errors and omissions.

4.1 Latest Software Enhancements

This section describes software enhancements that were implemented after the Oracle Rdb Version 7.0 documentation entered final production.

4.2 Additions and Changes to the Oracle Rdb Documentation for Version 7.0 and Earlier

This section provides information about late-breaking changes or other information that was missing or changed in the Oracle Rdb documentation for Version 7.0 and earlier releases.

4.2.1 Additions and Changes to the *Oracle Rdb7 and Oracle CODASYL DBMS: Guide to Hot Standby Databases* Documentation

OpenVMS VAX OpenVMS Alpha

The Hot Standby software has been enhanced regarding how it handles after-image journal files. Section 4.2.4 in the *Oracle Rdb7 and Oracle CODASYL DBMS: Guide to Hot Standby Databases* states the following information:

If an after-image journal switchover operation is suspended when replication operations are occurring, you must back up one or more of the modified after-image journals to add a new journal file.

This restriction has been removed. Now, you can add journal files or use the emergency AIJ feature of Oracle Rdb Release 7.0 to automatically add a new journal file. Note the following when adding an AIJ file versus adding an emergency AIJ file:

- You can add an AIJ file to the master database and it will be replicated on the standby database. If replication operations are active, the AIJ file is created on the standby database right away. If replication operations are not active, the AIJ file is created on the standby database when replication operations are restarted.
- Emergency AIJ files can be added anytime. If replication operations are active, the emergency AIJ file is created on the standby database right away. However, because emergency AIJ files are not journaled, starting replication after you create an emergency AIJ will fail. You cannot start replication operations because the Hot Standby software detects a mismatch in the number of after-image journal files on the master compared to the standby database.

If an emergency AIJ file is created on the master database when replication operations are not active, you must perform a master database backup and then restore the backup on the standby database. Otherwise, an AIJSIGNATURE error will result.

4.3 *Oracle Rdb7 SQL Reference Manual*

This section provides information that is missing from or changed in V7.0 of the *Oracle Rdb7 SQL Reference Manual*.

4.3.0.1 Reorganization of the *Oracle Rdb7 SQL Reference Manual*

Due to a page count restriction for hardcopy documentation, the *Oracle Rdb7 SQL Reference Manual* has been reorganized. Volume 2 now includes the ALTER DATABASE Statement through the DECLARE Variable Statement. Volume 3 now includes the DELETE Statement through the WHENEVER Statement.

4.3.1 Incorrect Qualifier for SQL Module Language Documented

The OpenVMS LC_PROC_NAMES qualifier for the SQL module language should be LOWERCASE_PROCEDURE_NAMES. This error has been fixed in the V7.0 *Oracle Rdb7 SQL Reference Manual*.

This error also appears in the *Migrating Oracle Rdb7 Databases and Applications to Digital UNIX*. This manual will be fixed in a future release.

4.3.2 Incorrect Digital UNIX Link Command for SQL Precompiler Documented

The section titled *SQL Precompiler Command Line for Digital UNIX* in the SQL Precompiler chapter of the *Oracle Rdb7 SQL Reference Manual* shows several examples incorrectly using the -lsql link command.

The correct link command is -lrdbsql for Oracle Rdb. See Section 3.2.1 for more information about this change.

This error will be corrected in the next published version of the *Oracle Rdb7 SQL Reference Manual*.

4.4 *Oracle RMU Reference Manual*

This section provides information that is missing from the 7.0 of the *Oracle RMU Reference Manual*.

4.4.1 New Transaction_Mode Qualifier for Some Oracle RMU Commands

A new qualifier, Transaction_Mode, has been added to the RMU Copy, Move_Area, Restore, and Restore Only_Root commands. You can use this qualifier to set the allowable transaction modes for the database root file created by these commands. If you are not creating a root file as part of one of these commands, for example, you are restoring an area, attempting to use this qualifier returns a CONFLSWIT error. This qualifier is similar to the SET TRANSACTION MODE clause of the CREATE DATABASE command in interactive SQL.

The primary use of this qualifier is when you restore a backup file (of the master database) to create a Hot Standby database. Include the Transaction_Mode qualifier on the RMU Restore command when you create the standby database (prior to starting replication operations). Because only read-only transactions are allowed on the standby database, you should use the Transaction_Mode=Read_Only qualifier setting. This setting prevents modifications to the standby database at all times, even when replication operations are not active.

You can specify the following transaction modes for the Transaction_Mode qualifier:

All
Current
None
[No]Batch_Update
[No]Read_Only
[No]Exclusive
[No]Exclusive_Read
[No]Exclusive_Write
[No]Protected
[No]Protected_Read
[No]Protected_Write
[No]Shared
[No]Shared_Read
[No]Shared_Write

Note that [No] indicates that the value can be negated. For example, the NoExclusive_Write option indicates that exclusive write is not an allowable access mode for this database. If you specify the Shared, Exclusive, or Protected option, Oracle RMU assumes you are referring to both reading and writing in these modes. For example, the Transaction_Mode=Shared option indicates that you want both Shared_Read and Shared_Write as transaction modes. No mode is enabled unless you add that mode to the list or you use the ALL option to enable all modes.

You cannot negate the following three options: All, which enables all transaction modes; None, which disables all transaction modes; and Current, which enables all transaction modes that are set for the source database. If you do not specify the Transaction_Mode qualifier, Oracle RMU uses the transaction modes enabled for the source database.

You can list one qualifier that enables or disables a particular mode followed by another that does the opposite. For example, Transaction_Mode=(NoShared_Write, Shared) is ambiguous because the first value disables Shared_Write access while the second value enables Shared_Write access. Oracle RMU resolves the ambiguities by first enabling all modes that are enabled by the items in the Transaction_Mode list and then disabling those modes that are disabled by items in the Transaction_Mode list. The order of items in the list is irrelevant. In the example discussed, Shared_Read is enabled and Shared_Write is disabled.

The following example shows how to set a newly restored database to allow read-only transactions only. After Oracle RMU executes the command, the database is ready for you to start Hot Standby replication operations.

```
$ RMU /RESTORE /TRANSACTION_MODE=READ_ONLY MF_PERSONNEL.RBF
```

4.4.2 RMU Server After_Journal Stop Command

If database replication is active and you attempt to stop the database AIJ Log Server, Oracle Rdb returns an error. You must stop database replication before attempting to stop the server.

In addition, a new qualifier, Output=filename, has been added to the RMU Server After_Journal Stop command. This optional qualifier allows you to specify the file where the operational log is to be created. The operational log records the transmission and receipt of network messages.

If you do not include a directory specification with the file name, the log file is created in the database root file directory. It is invalid to include a node name as part of the file name specification.

Note that all Hot Standby bugcheck dumps are written to the corresponding bugcheck dump file; bugcheck dumps are not written to the file you specify with the Output qualifier.

4.4.3 Incomplete Description of Protection Qualifier for RMU Backup After_Journal Command

Digital UNIX

The description of the Protection Qualifier for the RMU Backup After_Journal command is incomplete in the *Oracle RMU Reference Manaul for Digital UNIX*. The complete description is as follows:

The Protection qualifier specifies the system file protection for the backup file produced by the RMU Backup After_Journal command. If you do not specify the Protection qualifier, the default access permissions are `-rw-r-----` for backups to disk or tape.

Tapes do not allow delete or execute access and the superuser account always has both read and write access to tapes. In addition, a more restrictive class accumulates the access rights of the less restrictive classes.

If you specify the Protection qualifier explicitly, the differences in access permissions applied for backups to tape or disk as noted in the preceding paragraph are applied. Thus, if you specify Protection=(S,O,G:W,W:R), the access permissions on tape becomes `rw-rw-r--`. ♦

4.5 Changes to the *Oracle Rdb7 Guide to Database Performance and Tuning*

The following section provides corrected, clarified, or omitted information for the *Oracle Rdb7 Guide to Database Performance and Tuning* manual.

4.5.1 Error in Updating and Retrieving a Row by Dbkey Example

Example 3-22 in Section 3.8.3 that shows how to update and retrieve a row by dbkey is incorrect. The example show appear as follows:

```
SQL> ATTACH 'FILENAME MF_PERSONNEL.RDB';
SQL> --
SQL> -- Declare host variables
SQL> --
SQL> DECLARE :hv_row INTEGER;              -- Row counter
SQL> DECLARE :hv_employee_id ID_DOM;       -- EMPLOYEE_ID field
SQL> DECLARE :hv_employee_id_ind SMALLINT; -- Null indicator variable
SQL> --
SQL> DECLARE :hv_dbkey CHAR(8);            -- DBKEY storage
SQL> DECLARE :hv_dbkey_ind SMALLINT;       -- Null indicator variable
SQL> --
SQL> DECLARE :hv_last_name LAST_NAME_DOM;
SQL> DECLARE :hv_new_address_data_1 ADDRESS_DATA_1_DOM;
SQL> --
SQL> -- Set host variables
SQL> --
SQL> SET TRANSACTION READ WRITE;
SQL> BEGIN
cont> --
cont> -- Set the search value for SELECT
cont> --
cont>  SET :hv_last_name = 'Ames';
cont> --
cont> -- Set the NEW_ADDRESS_DATA_1 value
cont> --
cont> SET :hv_new_address_data_1 = '100 Broadway Ave.';
cont> END;
SQL> COMMIT;
SQL> --
SQL> SET TRANSACTION READ ONLY;
SQL> BEGIN
cont> SELECT E.EMPLOYEE_ID, E.DBKEY
cont>   INTO :hv_employee_id INDICATOR :hv_employee_id_ind,
cont>        :hv_dbkey INDICATOR :hv_dbkey_ind
cont>   FROM EMPLOYEES E
cont> WHERE E.LAST_NAME = :hv_last_name
cont> LIMIT TO 1 ROW;
cont> --
cont> GET DIAGNOSTICS :hv_row = ROW_COUNT;
cont> END;
SQL> COMMIT;
SQL> --
SQL> SET TRANSACTION READ WRITE RESERVING EMPLOYEES FOR SHARED WRITE;
SQL> BEGIN
cont> IF (:hv_row = 1) THEN
cont>    BEGIN
cont>    UPDATE EMPLOYEES E
cont>      SET E.ADDRESS_DATA_1 = :hv_new_address_data_1
cont>    WHERE E.DBKEY = :hv_dbkey;
cont>    END;
cont> END IF;
cont> END;
SQL> COMMIT;
SQL> --
SQL> -- Display result of change
SQL> --
SQL> SET TRANSACTION READ ONLY;
SQL> SELECT E.*
cont> FROM EMPLOYEES E
cont> WHERE E.DBKEY = :hv_dbkey;
 EMPLOYEE_ID   LAST_NAME       FIRST_NAME   MIDDLE_INITIAL
   ADDRESS_DATA_1              ADDRESS_DATA_2        CITY
      STATE   POSTAL_CODE   SEX    BIRTHDAY      STATUS_CODE
 00416         Ames            Louie        A
   100 Broadway Ave.                                 Alton
```

```
     NH      03809         M      13-Apr-1941   1

1 row selected
SQL>
```

The new example will appear in a future publication of the *Oracle Rdb7 Guide to Database Performance and Tuning* manual.

4.5.2 Error in Calculation of Sorted Index in Example 3-46

Example 3-46 in Section 3.9.5.1 shows the output when you use the RMU Analyze Indexes command and specify the Option=Debug qualifier and the DEPARTMENTS_INDEX sorted index.

The description of the example did not include the 8 byte dbkey in the calculation of the sorted index. The complete description is as follows:

The entire index (26 records) is located on pages 2 and 3 in logical area 72 and uses 188 bytes of a possible 430 bytes or the node record is 47 percent full. Note that due to index compression, the node size has decreased in size from 422 bytes to 188 bytes and the percent fullness of the node records has dropped from 98 to 47 percent. Also note that the used/avail value in the summary information at the end of the output does not include the index header and trailer information, which accounts for 32 bytes. This value is shown for each node record in the detailed part of the output. The number of bytes used by the index is calculated as follows: the sort key is 4 bytes plus a null byte for a total of 5 bytes. The prefix is 1 byte and the suffix is 1 byte. The prefix indicates the number of bytes in the preceding key that are the same and the suffix indicates the number of bytes that are different from the preceding key. The dbkey pointer to the row is 8 bytes. There are 26 data rows multiplied by 15 bytes for a total of 390 bytes. The 15 bytes include:

- 7 bytes for the sort key: length + null byte + prefix + suffix
- 8 bytes for the dbkey pointer to the row

Add 32 bytes for index header and trailer information for the index node to the 390 bytes for a total of 422 bytes used. Index compression reduces the number of bytes used to 188 bytes used.

The revised description will appear in a future publication of the *Oracle Rdb7 Guide to Database Performance and Tuning* manual.

Index