Oracle® Rdb for OpenVMS

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Oracle® Rdb for OpenVMS

Release Notes

Release 7.2.3.2

December 2008

Oracle Rdb Release Notes, Release 7.2.3.2 for OpenVMS

Contents

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb Release 7.2.3.2. The notes describe changed and enhanced features; upgrade and compatibility information; new and existing software problems and restrictions; and software and documentation corrections.

Intended Audience

This manual is intended for use by all Oracle Rdb users. Read this manual before you install, upgrade, or use Oracle Rdb Release 7.2.3.2.

Document Structure

This manual consists of the following chapters:

Chapter 1	Describes how to install Oracle Rdb Release 7.2.3.2.
Chapter 2	Describes problems corrected in Oracle Rdb Release 7.2.3.2.
Chapter 3	Describes problems corrected in Oracle Rdb Release 7.2.3.1.
Chapter 4	Describes problems corrected in Oracle Rdb Release 7.2.3.0.
Chapter 5	Describes enhancements introduced in Oracle Rdb Release 7.2.3.2.
Chapter 6	Describes enhancements introduced in Oracle Rdb Release 7.2.3.0.
Chapter 7	Provides information not currently available in the Oracle Rdb documentation set.
Chapter 8	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb Release 7.2.3.2.

Chapter 1
Installing Oracle Rdb Release 7.2.3.2

This software update is installed using the OpenVMS VMSINSTAL utility.

NOTE

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

1.1 Oracle Rdb on HP OpenVMS Industry Standard 64

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

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

1.2 Requirements

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

- This Oracle Rdb release requires the following OpenVMS environments:
  - ♦ OpenVMS Alpha V8.2 to V8.3−x.
  - ♦ OpenVMS Industry Standard 64 V8.2−1 to V8.3−x.
- Oracle Rdb must be shutdown before you install this update kit. That is, the command file SYS$STARTUP:RMONSTOP72.COM should be executed before proceeding with this installation. If you have an OpenVMS cluster, you must shutdown the Rdb Release 7.2 monitor on all nodes in the cluster before proceeding.
- The installation requires approximately 280,000 blocks for OpenVMS Alpha systems.
- The installation requires approximately 500,000 blocks for OpenVMS I64 systems.
- Oracle strongly recommends that all available OpenVMS patches are installed on all systems prior to installing Oracle Rdb. Contact your HP support representitive for more information and assistance.

1.3 Intel Itanium Processor 9100 "Montvale" Support

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

1.4 Maximum OpenVMS Version Check

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

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

1.5 Database Format Changed

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

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

1.6 Using Databases from Releases Earlier than V7.0

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

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

1.7 Invoking the VMSINSTAL Procedure

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

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

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

```
@SYS$UPDATE:VMSINSTAL RDBV72320IM device−name
```

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

```
@SYS$UPDATE:VMSINSTAL RDBV72320AM device−name
```

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

```
@SYS$UPDATE:VMSINSTAL RDBV72321AM device−name
```

device−name

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

1.8 Stopping the Installation

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

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

1.9 After Installing Oracle Rdb

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

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

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

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

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

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

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

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

1.10 VMS$MEM_RESIDENT_USER Rights Identifier Required

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

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

1.11 Installation, Configuration, Migration, Upgrade Suggestions

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

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

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

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

Table 1−1 Migration Suggestions

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

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

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

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

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

Chapter 2
Software Errors Fixed in Oracle Rdb Release 7.2.3.2

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.3.2.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 Result Data Type From NEGATE was Incorrect

Exact numeric type (TINYINT, SMALLINT and INTEGER) have an interesting property. The most negative value cannot be negated and still be represented by the same number of bits. For example, a TINYINT can hold the value −128, but the negated value (128) requires a SMALLINT. Similarly, a SMALLINT type can hold −32768 but the negated value (32768) requires an INTEGER, and so on.

Prior releases of Oracle Rdb failed to allocate a sufficiently large data type for the result of the negate operation which could lead to integer overflow errors.

This problem has been corrected in Oracle Rdb Release 7.2.3.2. Negate, and related operations such as the ABS function, will now result in the correct type.

2.1.2 Floating Multiply May Produce Lower Precision Result

Bug 7383036

Oracle Rdb produces a floating point result whenever a divide operation (/) is used in a SQL or RDO query. It has been reported that in some cases only involving scaled numeric data that this floating result, when multiplied by a value and then converted back to the source data type, may contain a significant rounding error on Alpha hardware. This was obvious when comparing the results from OpenVMS systems running on VAX or Integrity hardware.

This release of Oracle Rdb has restructured the generated code for Alpha systems so that this operation maintains a larger magnitude value during the multiply (delaying the scaling of the integer value for a short time).

This change affects TINYINT, SMALLINT, INTEGER and BIGINT with scale value greater than 0. In most cases, there should be little difference in the results but, in some cases, the change of operation order will be noticeable because it preserves extra bits of accuracy.

This problem has been corrected in Oracle Rdb Release 7.2.3.2.

2.1.3 Zero−Length String Compare on Itanium Using Run−Time Native Compiler Could Cause Bugcheck

Bug 7457310

When using the optional run−time native code generator feature (enabled by default starting with Oracle Rdb Release 7.2.3), certain zero−length string compares could result in incorrect code being generated leading to a bugcheck at runtime due to an ACCVIO from the generated code.

The following example demonstrates one possible cause of this problem:

```
SQL> CREATE DATABASE FILENAME TESTDB;
SQL> CREATE TABLE T1 (C1 CHAR(7));
SQL> INSERT INTO T1 (C1) VALUES ('1234567');
1 row inserted
SQL> INSERT INTO T1 (C1) VALUES ('7654321');
1 row inserted
SQL> COMMIT;
SQL> SELECT DISTINCT C1,'' FROM T1;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DUA0:[USER]RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DUA0:[USER]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=0000000001A54000, PC=0000000000641120, PS=0000001B
```

In this example, the bugcheck dump "footprint" would be similar to:

```
***** Exception at 00000000816D2040 : symbol not found
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000001A54000, PC=00000000816D2040, PS=0000000B
Saved PC = 00000000809BFE30 : RDMSHRP721\RDMS$$EXE_NEXT + 00005E10
Saved PC = 00000000809C1820 : RDMSHRP721\RDMS$$EXE_NEXT + 00007800
Saved PC = 0000000080AA8A20 : RDMSHRP721\RDMS$$C_EXE_NEXT + 00000080
Saved PC = 0000000080373D20 : RDMSHRP721\RDMS_EXE_INTERP + 0000FCD0
```

As a workaround for this problem, the run−time native compiler could be disabled by defining the logical name RDMS$BIND_CODE_OPTIMIZATION to the value "0" or setting the debug flag CODE_OPTIMIZATION to zero.

This problem has been corrected in Oracle Rdb Release 7.2.3.2. The run−time native compiler now generates correct code for zero−length string comparisons.

2.1.4 Query Slows Down Using Seq Scan After Reaching ThreLim

Bug 7384420

The simple select query slows down dramatically when the cardinality of the table hits 1651910465 and the strategy switches to "Fin Seq" after the BgrNdx1 reaches "ThreLim".

```
select * from T1 where a1=5;
Tables:
  0 = T1
Leaf#01 FFirst 0:T1 Card=1651910465
  Bool: 0.A1 = 5
  BgrNdx1 U1 [1:1] Fan=14
    Keys: 0.A1 = 5
~Estim  U1 Sorted: Split lev=3, Seps=1 Est=222
~E#0003.01(1) Estim   Index/Estimate 1/222
~E#0003.01(1) BgrNdx1 ThreLim  DBKeys=0  Fetches=2+0  RecsOut=0
~E#0003.01(1) FgrNdx  FFirst   DBKeys=0  Fetches=0+0  RecsOut=0*`ABA
          A1            A2            A3
           5             1             1
           5             2             1
...etc...
           5          2047             1
~E#0001.01(1) Fin  Seq   DBKeys=1659910464  Fetches=0+3277523  RecsOut=2048
           5          2048             1
2048 rows selected
```

```
show stat

                  process statistics at 10-OCT-2008 11:04:46.45
         elapsed time =   0 00:52:23.97              CPU time =   0 00:50:34.64
     page fault count = 41               pages in working set = 19984
   buffered I/O count = 2133                 direct I/O count = 3277555
      open file count = 13               file quota remaining = 1987
           locks held = 159                   locks remaining = 31841
      CPU utilization = 96.5%              AST quota remaining = 995
```

If one row is deleted from the table, the query runs fast (see following example).

```
delete from T1 where A1 = 1671910465;
commit;
select * from T1 where a1=5;
Tables:
  0 = T1
Leaf#01 FFirst 0:T1 Card=1651910465
  Bool: 0.A1 = 5
  BgrNdx1 U1 [1:1] Fan=14
    Keys: 0.A1 = 5
~Estim  U1 Sorted: Split lev=3, Seps=1 Est=222
~E#0004.01(1) Estim   Index/Estimate 1/222
          A1            A2            A3
           5             1             1
        ...etc...
           5          1023             1
~E#0001.01(1) BgrNdx1 EofBuf   DBKeys=1024  Fetches=2+1  RecsOut=1024
~E#0001.01(1) FgrNdx  FFirst   DBKeys=1024  Fetches=0+3  RecsOut=1024`ABA
~E#0001.01(1) BgrNdx1 EofData  DBKeys=2048* Fetches=0+0  RecsOut=1024 #Bufs=5
           5          1024             1
        ...etc...
           5          2047             1
~E#0002.01(1) Fin     TTbl     DBKeys=2048  Fetches=0+0  RecsOut=2048
           5          2048             1
2048 rows selected
```

The problem is caused by the overflow of the index threshold limit in BgrNdx1, where the background index reaches the threshold limit (ThreLim), abandons the foreground FgrNdx (FFirst), and finally resolves to the sequential tactic "Fin Seq".

The cardinality value of 1671910465 causes an overflow in the threshold limit of the index internally.

If the cardinality value is reduced by just one, the BgrNdx1 successfully scans the background index, switches to FgrNdx (FFirst) after it hits the end of buffer (EofBuf) at 1024 dbkeys, and finally delivers the rows from the background index.

This problem has been corrected in Oracle Rdb Release 7.2.3.2.

2.1.5 String Compares of More Than 255 Bytes Could Return False Matches

Bug 7560946

When using the optional run−time native code generator feature (enabled by default starting with Oracle Rdb Release 7.2.3), certain compares of fixed−length strings greather than 255 bytes could result in incorrect code

being generated leading to possible false matches if the leading bytes of the string were equal.

As a workaround for this problem, the run−time native compiler could be disabled by defining the logical name RDMS$BIND_CODE_OPTIMIZATION to the value "0" or setting the debug flag CODE_OPTIMIZATION to zero.

This problem has been corrected in Oracle Rdb Release 7.2.3.2. The run−time native compiler now generates correct code for string comparisons of any length.

2.2 SQL Errors Fixed

2.2.1 Incorrect COMPRESSION Attributes Applied to Vertical Row Partitioned Table

Bug 7278508

In prior releases of Oracle Rdb, the CREATE STORAGE MAP statement that used the COLUMNS clause to create a vertical row partitioned table would not apply the correct compression to each partition.

The following example shows that the CREATE STORAGE MAP specifies that only the last partition be compressed but the created map has all partitions with compression enabled.

```
SQL> create storage map t_m for t
cont>     store columns (i,tx)
cont>         disable compression
cont>         in a1
cont>     store columns (i2,tx2)
cont>         disable compression
cont>         in a2
cont>     store columns (tx3)
cont>         enable compression
cont>         in a3
cont> ;
SQL> show storage map (partition) t_m;
     T_M
 For Table:             T
 Partitioning is:       UPDATABLE
 Store clause:          STORE columns (i,tx)
        disable compression
        in a1
    store columns (i2,tx2)
        disable compression
        in a2
    store columns (tx3)
        enable compression
        in a3

 Partition information for storage map:

 Vertical Partition: VRP_P001
 Compression is:        ENABLED
  Partition: (1) SYS_P00061
   Storage Area: A1

 Vertical Partition: VRP_P002
 Compression is:        ENABLED
  Partition: (1) SYS_P00062
   Storage Area: A2

 Vertical Partition: VRP_P003
 Compression is:        ENABLED
  Partition: (1) SYS_P00063
   Storage Area: A3
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.3.2. Existing tables can still be accessed. After this release has been installed, the table can be unloaded, the storage map re−created and the data reloaded to achieve the desired compression characteristics.

2.2.2 Unexpected Bugcheck During ALTER INDEX ... BUILD PARTITION on Sorted Ranked Index

Bug 7360420

In prior releases of Oracle Rdb, it was possible that ALTER INDEX could bugcheck with a footprint similar to the one shown below. This occurred while building the bitmap for the SORTED RANKED index.

- PSIIBUILD2BUILDFROMBOTTOM + 00000A2C
- PSII2CREATETREE + 00000244
- RDMS$$KOD_CREATE_TREE + 000001E4
- RDMS$$KOD_CREATE_INDEX + 000005DC
- RDMS$$CREATE_INDEX_INFO + 00004CFC
- RDMS$$RELEASE_DDL_VM_HNDLR + 00000A24
- BLI$CALLG + 000000BC
- KOD$SETSTK_AND_CONTINUE + 0000019C

This problem can be avoided by using either CREATE INDEX or ALTER INDEX ... BUILD ALL PARTITIONS.

This problem has been corrected in Oracle Rdb Release 7.2.3.2.

2.2.3 Change in Handling of Dynamic Cursor Names

In prior versions of Oracle Rdb, cursor names passed to the DECLARE CURSOR statement (extended dynamic) were implicitly upper−cased. However, we have observed considerable CPU overhead for this action. Interfaces such as Oracle SQL/Services, OCI Services for Rdb, Native JDBC Driver, and so on, all use a consistent naming scheme for cursors and do not benefit from the upper−casing of the name. Therefore, with this release of Oracle Rdb, we no longer uppercase the cursor name passed to the dynamic SQL statement.

Applications will typically create and use a consistent case for cursor names and should see no change in behavior from this performance change to Dynamic SQL.

2.2.4 Unexpected DISABLED Indices After Database Created by IMPORT DATABASE Statement

Bug 7578428

The ALTER INDEX ... BUILD ALL PARTITIONS statement implicitly marks the index as MAINTENANCE IS ENABLED IMMEDIATE. Unfortunately, if the index was originally marked as MAINTENANCE IS DISABLED, a subsequent EXPORT DATABASE and IMPORT DATABASE would recreate the index again marked as MAINTENANCE IS DISABLED.

The following example shows this sequence. Note that the index after the import is marked as ***Index is no longer maintained***.

```
SQL> create table my_tab (my_col1 integer, my_col2 integer);
SQL> insert into  my_tab value (1,1);
1 row inserted
SQL> create index my_ndx on my_tab (my_col2);
SQL> alter index my_ndx maintenance is disabled;
SQL> alter index my_ndx truncate all partitions;
%RDB-W-META_WARN, metadata successfully updated with the reported warning
-RDMS-W-IDXBLDPEND, index in build pending state - maintenance is disabled
SQL> alter index my_ndx build all partitions;
SQL> show index *
User indexes in database with filename testdb
Indexes on table MY_TAB:
MY_NDX                          with column MY_COL2
  Duplicates are allowed
  Type is Sorted
  Key suffix compression is DISABLED
SQL> commit;
SQL> disconnect all;
SQL>
SQL> export database filename testdb into save_testdb;
SQL> drop data filename testdb;
SQL> import database from save_testdb filename testdb;
SQL>
SQL> show index *
User indexes in database with filename testdb
Indexes on table MY_TAB:
MY_NDX                          with column MY_COL2
  Duplicates are allowed
  Type is Sorted
  Key suffix compression is DISABLED
  Index is no longer maintained
SQL>
```

This problem is caused by stale metadata (in the Rdb$EXTENSION_PARAMETERS) column for the index.

To avoid this problem, the index maintenance can be changed as follows:

1. Name the partitions for the index to make it easier to specify these names in the various ALTER INDEX commands.

   ```
   SQL> create index MY_NDX2 on MY_TAB (my_col2)
   cont> store in RDB$SYSTEM (partition MY_NDX2);
   ```

   For an existing index, an ALTER INDEX ... RENAME PARTITION ... statement can be used to override the Rdb-generated names for each partition.
2. Replace the BUILD ALL PARTITIONS action with a BUILD PARTITION for each partition.

   ```
   SQL> alter index MY_NDX2 build partition MY_NDX2;
   %RDB-W-META_WARN, metadata successfully updated with the reported warning
   -RDMS-W-IDXBLDPEND, index in build pending state - maintenance is disabled
   ```
3. Finally, execute the ALTER INDEX command to set MAINTENANCE as ENABLED IMMEDIATE. This command correctly removes the stale metadata.

   ```
   SQL> alter index MY_NDX2 maintenance is enabled immediate;
   ```

This problem has been corrected in Oracle Rdb Release 7.2.3.2. ALTER INDEX now removes the stale setting from the index allowing EXPORT and IMPORT to process the index correctly.

2.3 RMU Errors Fixed

2.3.1 Use of External Media Manager or Tape Librarian Causes a Hang or an ACCVIO

An RMU command can exit with an ACCVIO or hang, especially when the command is using a media manager or tape librarian. In some cases, the per thread stack might have been exhausted and be the cause of such a hang or ACCVIO. But there could be other reasons for this situation as well.

This modification adds a no−access virtual memory page to either side of a per thread stack. This causes an ACCVIO if a thread tries to use stack space beyond the stack limits.

Here is how to find out whether or not the ACCVIO was caused by a stack size too small. Repeat the command with the following logicals defined to see the current stack limits:

```
$ CREATE/NAME_TABLE COSI_ENV_USER
$ DEFINE/TABLE=COSI_ENV_USER RMU$DEBUG_FLAGS −
      "RMU_DEBUG_TIME,RMU_DEBUG_ANY"
$ RMU...
15:59:52.80: BF  thread DIO: X00C86010 length: X0138 STKLIM: X00C8A000:X00C92000
15:59:52.80: DBF thread DIO: X00C94160 length: X0140 STKLIM: X00C98000:X00CA0000
15:59:52.80: DBF thread DIO: X00CA22B0 length: X0140 STKLIM: X00CA6000:X00CAE000
%SYSTEM−F−ACCVIO, access violation, reason mask=01, −
                  virtual address=0000000000C89FB0, PC=...
```

The example shows the virtual address of the ACCVIO 00C89FB0 is just below the low stack limit 00C8A000 and within the guard page. This indicates a possible stack size problem and can be avoided by increasing the stack size per thread.

To add extra stack space, define the following logical:

```
$ DEFINE RDM$BIND_LIBRARIAN_ADDITIONAL_STACK 10240

...adds another 10KB stack space per RMU thread.
```

This logical can be used even if no media manager or tape librarian is being used.

This problem has been corrected in Oracle Rdb Release 7.2.3.2.

2.3.2 RMU/BACKUP Sometimes Dismounts Last Volume in a Multi−tape Drive Case

When running an RMU Backup using multiple tape drives, the last volume is sometimes dismounted.

In a multi−tape drive case, all volumes except the last one are so−called continuation volumes and cannot be appended to. Hence these volumes are dismounted at the end of the backup. The last volume can still be appended to and is therefore not to be dismounted.

This was caused by a false setting of a last volume flag. This problem has been corrected in Oracle Rdb Release 7.2.3.2.

2.3.3 Parallel Backup Hangs

Bug 6628917

A parallel backup can hang depending on the number and size of storage areas and the number of tape drives being used.

This is caused by a feature in RMU Backup that waits for the first worker process to save the system area before other worker processes can proceed. In some cases, the system area was not assigned to executor process WORKER_001 and therefore it caused a hang.

As a workaround, the plan file can be edited and the line with the system area can be moved in front of the areas assigned to WORKER_001.

```
Example of PLAN file listing the system area RDB$SYSTEM first in the
storage area list for WORKER_001:

Executor Parameters :
    Executor Name = WORKER_001
    Executor Type = Worker
    Start Storage Area List
        RDB$SYSTEM,
        A0010,
        ...
```

With this change, the system area is always assigned to the top of the list for WORKER_001.

This problem has been corrected in Oracle Rdb Release 7.2.3.2.

2.3.4 Users Were Audited Who Held Identifiers Not Enabled For Auditing

Bug 6531721

If you specify the command

```
$ RMU/SET AUDIT=IDENTIFIER=(X) database
```

identifier X is added to a list of enabled user identifiers whose access to an Oracle Rdb database can be audited. If a user does not hold an identifier that matches this list, then Oracle Rdb will not send audit entries to VMS for the database user when auditing is started. The following command shows the identifiers currently enabled for audit.

```
$ RMU/SHOW AUDIT/IDENTIFIERS database
```

Because of a problem checking user identifiers against the list of identifiers currently enabled for auditing, users of Rdb databases who did not hold one of these identifiers were audited. This problem has been fixed and now only users holding enabled identifiers will be audited when they access Rdb databases.

The following example of this problem shows that although auditing is enabled only for "USERA", if, when auditing is started, another user accesses the database who holds an identifier which has not been enabled for auditing, auditing entries for that user have been created when they should not have been.

```
$ RMU/SET AUDIT/ENABLE=IDENTIFIER=(USERA) MF_PERSONNEL
$ RMU/SET AUDIT/ENABLE=DACCESS=DATABASE/PRIVILEGES=(ALL) MF_PERSONNEL
$ RMU/SET AUDIT/ENABLE=(DACCESS,PROTECTION,RMU) MF_PERSONNEL
$ RMU/SET AUDIT/EVERY MF_PERSONNEL
$ RMU/SET AUDIT/FLUSH MR_PERSONNEL
$ RMU/SET AUDIT /TYPE=AUDIT /START MF_PERSONNEL
$ RMU/SET AUDIT/ENABLE=DACCESS=TABLE=(EMPLOYEES) -
   /PRIVILEGES=(ALTER, CREATETAB, DBADM, DBCTRL, DELETE, DROP, INSERT, -
   SECURITY, UPDATE, FAILURE) MF_PERSONNEL
$ RMU/SHOW AUDIT/IDENTIFIERS MF_PERSONNEL
Enabled identifiers:
    (IDENTIFIER=[RDB,USERA])

$ SQL
ATT 'FILENAME MF_PERSONNEL';
SELECT DISTINCT AUDIT$TIME_STAMP,AUDIT$OPERATION,AUDIT$USER_NAME,
AUDIT$OBJECT_NAME FROM AUDIT_RECORDS -
WHERE AUDIT$USER_NAME = 'USERB';
 AUDIT$TIME_STAMP
   AUDIT$OPERATION                    AUDIT$USER_NAME
      AUDIT$OBJECT_NAME
 22-SEP-2008 11:48:03.56
   Attach Database                    USERB
      DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
      >>
      >>
      >>
 22-SEP-2008 11:48:20.45
   Insert Record                      USERB
      EMPLOYEES
      >>
      >>
      >>
 22-SEP-2008 11:48:51.71
   Delete Record                      USERB
      EMPLOYEES
      >>
      >>
      >>
3 rows selected
```

There is no workaround to avoid this problem except to only display audit entries in SQL for named users.

This problem has been corrected in Oracle Rdb Release 7.2.3.2.

2.3.5 RMU−W−AIJBCKFAIL − RMU/RECOVER Reports Incomplete AIJ Backup

Using /FORMAT=NEW_TAPE to backup and restore a journal file may report a warning at the end of the recover operation.

```
$ RMU /RESTORE /NOLOG /NOCDD /AFTER MFP.RBF
$ RMU /RECOVER /NOLOG MFP.ABF /FORMAT=NEW_TAPE
...
```

```
%RMU-W-AIJBCKFAIL, the AIJ backup that created the AIJ file did not complete
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence number
needed will be 0
```

This would happen when an RMU/BACKUP/AFTER_JOURNAL command would exactly fill an internal buffer when saving the last AIJ record. An internal block counter was not correctly incremented and hence the final AIJ_END record was tagged with a duplicate block number. The problem remained undetected during the save operation. All AIJ records were restored during a recover operation. However, the reported next sequence number was incorrect which could prevent a further recovery operation.

This problem has been corrected in Oracle Rdb Release 7.2.3.2.

2.4 LogMiner Errors Fixed

2.4.1 LogMiner Incorrectly Returns DBK Multiple Times Within Transaction

Bug 7532856

Starting with Oracle Rdb Release 7.2.3, the RMU /UNLOAD /AFTER_JOURNAL command could, in rare cases, return multiple versions of a row with the same database key within one transaction (this case is not allowed; only the final content of a row is to be returned within a transaction). The cause of the problem was related to an incorrect internal sort comparison not correctly handling duplicate database key removal.

This problem has been corrected in Oracle Rdb Release 7.2.3.2.

Chapter 3
Software Errors Fixed in Oracle Rdb Release 7.2.3.1

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.3.1.

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 Page Transfer Via Memory Lost Database Update

Bug 7243961

In very rare cases, when using the "Page Transfer Via Memory" feature, a database recovery (DBR) process may incorrectly avoid re−applying a database update. The conditions leading to this situation are complex and rare.

As a workaround, users of the "Page Transfer Via Memory" feature may wish to consider disabling it.

This problem has been corrected in Oracle Rdb Release 7.2.3.1. The database recovery (DBR) process starts applying updates from the correct point in the AIJ file.

3.1.2 Rollback Performance Improvement with Row Cache

Performance of ROLLBACK operations has been improved when the Row Cache feature is enabled for a database. Previously, all working set entries in all caches for the user were evaluated even if the transaction did not modify any cache or perform any cache latch operations.

This problem has been corrected in Oracle Rdb Release 7.2.3.1. Working set entries are only checked if a transaction has modified any cache or performed any cache latching operations.

3.1.3 Wrong Result from Query with Shared NULL Conjunct in OR Predicate

Bug 7254564

The following query returns the wrong result (100 rows instead of 0 rows).

```
set flags 'strategy,detail';
select employee_id, first_name, last_name
from employees e1
where
 (employee_id is not null and
  (employee_id is null or
   not exists (select * from employees e2
                where e2.employee_id = e1.employee_id
                    and e2.last_name = e1.last_name)));
Tables:
  0 = EMPLOYEES
  1 = EMPLOYEES
Cross block of 2 entries
  Cross block entry 1
    Leaf#01 FFirst 0:EMPLOYEES Card=100
      Bool: NOT MISSING (0.EMPLOYEE_ID)
      BgrNdx1 EMP_EMPLOYEE_ID [0:1] Fan=17
```

```
        Keys: NOT MISSING (0.EMPLOYEE_ID)
  Cross block entry 2
    Aggregate-F1: 0:COUNT-ANY (<subselect>)
    Conjunct: 1.LAST_NAME = 0.LAST_NAME
    Get     Retrieval by index of relation 1:EMPLOYEES
      Index name  EMPLOYEES_HASH [1:1]       Direct lookup
        Keys: 1.EMPLOYEE_ID = 0.EMPLOYEE_ID
        Bool: NOT MISSING (1.EMPLOYEE_ID)
 EMPLOYEE_ID   FIRST_NAME   LAST_NAME
 00164         Alvin        Toliver
 ...etc...
100 rows selected
```

In Rdb Release 7.2−200, the same query returns 0 rows.

```
Tables:
  0 = EMPLOYEES
  1 = EMPLOYEES
Cross block of 2 entries  Q1
  Cross block entry 1
    Leaf#01 FFirst 0:EMPLOYEES Card=100
      Bool: NOT MISSING (0.EMPLOYEE_ID)
      BgrNdx1 EMP_EMPLOYEE_ID [0:1] Fan=17
        Keys: NOT MISSING (0.EMPLOYEE_ID)
  Cross block entry 2
    Conjunct: MISSING (0.EMPLOYEE_ID) OR (<agg0> = 0)
    Aggregate-F1: 0:COUNT-ANY (<subselect>) Q2
    Conjunct: 1.LAST_NAME = 0.LAST_NAME
    Get     Retrieval by index of relation 1:EMPLOYEES
      Index name  EMPLOYEES_HASH [1:1]       Direct lookup
        Keys: 1.EMPLOYEE_ID = 0.EMPLOYEE_ID
        Bool: NOT MISSING (1.EMPLOYEE_ID)
0 rows selected
```

This problem is caused by the NULL conjunct in one of the operands in the OR predicate, shared by the NOT NULL conjunct at the outer leg of a cross strategy.

The key parts of this query which contributed to the error are:

1. The WHERE clause contains a NULL conjunct as one of the operands in an OR predicate which is an operand to an AND predicate with the other operand being a NOT NULL conjunct on the same column.
2. The other operand of the OR predicate is a NOT EXISTS clause checking the existence of the joined operation between the same two tables.

This problem has been corrected in Oracle Rdb Release 7.2.3.1.

3.2 SQL Errors Fixed

3.2.1 Unexpected SQL−F−DATTYPUNK Error from Dynamic SQL Applications

Bug 7249833

In Oracle Rdb Release 7.2.3, a problem was introduced in the handling of parameters to Dynamic SQL states for DECODE and the Simple CASE expression. Dynamic fails to assign a data type in some cases and reported a SQL−F−DATTYPUNK error. The following example is run using a Dynamic SQL testing tool and shows the reported problem.

```
$       RUN TEST$TOOLS:TESTER
attach 'filename PERSONNEL';
set dialect 'oracle level2';
select E.LAST_NAME, E.FIRST_NAME
from EMPLOYEES E
where EMPLOYEE_ID = DECODE(:b1, NULL ,E.EMPLOYEE_ID, :b2);
Error −1:
%SQL−F−DATTYPUNK, Data type unknown. Expression cannot use only host variables
select E.LAST_NAME, E.FIRST_NAME
from EMPLOYEES E
where EMPLOYEE_ID = (case :b1 when NULL then E.EMPLOYEE_ID else :b2 end);
Error −1:
%SQL−F−DATTYPUNK, Data type unknown. Expression cannot use only host variables
```

This problem has been corrected in Oracle Rdb Release 7.2.3.1.

3.2.2 Updated Support for SQL Pseudo Types for C Language

Bug 7248505

In prior versions of Oracle Rdb, SQL date/time types were supported by the SQL precompiler for C using a struct of two long fields (l0,l1). This caused informational errors from the C compiler when using /NOMEMBER_ALIGNMENT. In this release, Oracle Rdb has been changed to no longer generate this struct and instead use an int64 variable for SQL date/time values declared using SQL_TIMESTAMP, SQL_TIME, SQL_DATE_ANSI, SQL_DATE_VMS, and SQL_INTERVAL pseudo type definitions.

The following example shows the error previously generated by the C compiler.

```
$ sql$pre/cc=decc b /nomember_alignment/warn=enable=alignment

    long l0,l1;
.........^
%CC−I−MISALGNDSTRCT, This member requires longword alignment for efficient
access, but is contained in a struct containing byte alignment.  Consider
using #pragma nomember_alignment longword.
....
```

This problem has been corrected in Oracle Rdb Release 7.2.3.1.

Note

Although the internal layout was not documented nor was the struct supported for customer use by Oracle, it is possible that existing applications reference the fields of the struct. If this is the case, then it is possible to redefine the struct for use in applications. Include this definition at the top of the C module that requires the field references.

```
#pragma member_alignment save
#pragma member_alignment
typedef struct {
    long l0,l1;
} SQL_DATE_ANSI_X;
#define SQL_DATE_ANSI SQL_DATE_ANSI_X
#pragma member_alignment restore


The macro will be used to replace the definition generated by the SQL
precompiler.
```

3.2.3 Unexpected SQL−F−NUMXPREXP Error While Processing ABS Function

In prior releases of Oracle Rdb V7.2, the ABS function used with a parameter marker in Dynamic SQL could return the SQL−F−NUMXPREXP error. The following example shows such an example:

```
select * from rdb$database where rdb$flags = ABS(?);
Error −1:
%SQL−F−NUMXPREXP, A numeric expression was expected
```

When SQL cannot determine the type of the ABS function source, it should default to DOUBLE PRECISION as shown in this example.

```
select * from rdb$database where rdb$flags = ABS(?);
[1 fields]
 0//G Float: 0.1e0
```

This problem has been corrected in Oracle Rdb Release 7.2.3.1.

3.2.4 Unexpected Truncation of Application Parameter Data in Dynamic SQL

Bug 1523789

In prior releases of Oracle Rdb, Dynamic SQL would set the data type of a parameter marker based on context. Often this data type might have been resolved from a string literal and in many cases this string literal may have been smaller than the source application parameter. This resulted in truncation of the input and might have led to incorrect query results. Consider this simple example:

```
select * from PORTS where city = ? and ? <> 'Boston';
```

In this example, the second parameter was described in the SQLDA as CHAR(6) inheriting the length of the string literal, but the input source from the application was really a CHAR(20). This resulted in application data being truncated when passed to the SQL query.

This problem has been corrected in Oracle Rdb Release 7.2.3.1. In this release, any parameter which derives its data type from a string literal will be promoted to VARCHAR(2000) to ensure sufficient space is available for application parameter data.

3.2.5 Unexpected Prompt When Adding a Comment to an Index

In prior versions of Oracle Rdb, the ALTER INDEX statement tried to prevent unintended changes to the STORE clause by confirming this action with the prompt "This index was previously specified with a STORE clause. Continue? [N]". However, this prompt was incorrectly issued when only the COMMENT IS clause was used.

The following example shows the problem.

```
SQL> ALTER INDEX foo COMMENT 'SUPPORTS CONSTRAINT bar';
This index was previously specified with a STORE clause. Continue? [N]N
%SQL-F-CHGINDMAPSTP, Terminating operation at user's request
```

The workaround is to use the COMMENT ON INDEX statement or answer Y to the prompt.

This problem has been corrected in Oracle Rdb Release 7.2.3.1.

3.3 RMU Errors Fixed

3.3.1 RMU/BACKUP Exits with ACCVIO in ENCRYPTSHR

An RMU/BACKUP/ENCRYPT can fail with an ACCVIO with the following pattern in the RMUBUGCHK dump:

```
***** Exception at 0000000000F5C730 : Image ENCRYPT$ALG$AES + 00032730
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=0000000000BE6000, PC=0000000000F5C730, PS=0000001B
Saved PC = 0000000000E9AAEC : Image ENCRYPSHR + 00082AEC
Saved PC = 0000000000393808 : RMU72\RMUIO$CRYPT_INIT_BCK + 000002F8
```

The problem was caused by a buffer overrun in OpenVMS ENCRYPTSHR code. The problem has been reported to OpenVMS Engineering.

This problem has been corrected in Oracle Rdb Release 7.2.3.1. The RMU code has been modified to protect against the buffer overrun.

3.4 RMU Show Statistics Errors Fixed

3.4.1 RMU/SHOW STATISTICS Does Not Show BLOB Areas On Logical Area Information Screen

Bug 7242685

When selecting BLOB logical areas on the "Logical Area Overview" display, if system logical areas have not been enabled, BLOB logical areas will not be displayed. If the tools main item "Display all logical areas" is used, then the BLOB logical areas will be displayed.

This problem has been corrected in Oracle Rdb Release 7.2.3.1. When selecting BLOB logical areas, they will be displayed regardless of the "Display all logical areas" setting.

Chapter 4
Software Errors Fixed in Oracle Rdb Release 7.2.3.0

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.3.0.

4.1 Software Errors Fixed That Apply to All Interfaces

4.1.1 Inability to Cause Free Space Scan for First Insert After Attach

Bug 6807755

Oracle Rdb Release 7.2 introduced a mechanism that allows database users on the same cluster node to share information regarding the availability of free space. This optimization helps reduce cases of unexpected storage area growth when one process is inserting into a table while another process on the same node is deleting from the same table.

In a cluster environment, however, there is no cross−node sharing regarding the availability of free space within a logical area. In prior releases of Oracle Rdb, a periodic detach and attach could be used to force a process to initially scan for the next available free space when inserting a record. However, this technique would not always work with Oracle Rdb Release 7.2 due to the shared insert location.

This problem has been corrected in Oracle Rdb Release 7.2.3. In a cluster environment, a process will ignore the shared information regarding the availability of free space for the process's first insert into a logical area.

4.1.2 Insert ACCVIO Bugcheck on IA64

Bug 6820142

An ACCVIO bugcheck could occur during an insert into a table with many constraints. This was only an issue on Itanium and also only when running in "Interpreter Mode", which was the default for versions prior to 7.2.3.

```
***** Exception at FFFFFFFF80004090 : symbol not found
%SYSTEM−F−ACCVIO, access violation, reason mask=04, virtual address=
FFFFFFFFFFFF8870, PC=FFFFFFFF80004090, PS=0000000B
Saved PC = 0000000080328710 : RDMSHRP731\RDMS_EXEINT_CREATE_IKEY + 000000F0
Saved PC = 00000000803828C0 : RDMSHRP731\RDMS_EXE_INTERP + 000446F0
```

A possible workaround is to use the Run−Time Routine Native Compiler which is enabled by default starting in this release, Release 7.2.3.

Please see the "Enhancements and Changes Provided in Oracle Rdb Release 7.2.3" Chapter. The first entry is entitled "Optional Run−Time Routine Native Compiler on I64" and gives more details about how to use this feature.

This problem has been corrected in Oracle Rdb Release 7.2.3.

4.1.3 Processes With Multiple Database Attaches Stall "Waiting for RTUPB list" on IA64

Bug 6707976

In rare cases, a process with multiple streams (meaning attached to databases multiple times) could get into a condition where one stream needed to execute and was able to run but another stream was hibernating. A change in implementation during the IA64 port resulted in the hibernation happening in the stream's context rather than in the internal Rdb thread scheduler (as it would have been on Alpha). This problem has been corrected in Oracle Rdb Release 7.2.3.

4.1.4 Multi−Threaded Processes Hang in HIB Status

Bug 6707976

In some cases, multi−threaded processes may appear to hang in a hibernating state when using Oracle Rdb. This condition can be due to both Oracle Rdb and the OpenVMS user−mode process thread scheduler using hiber/wake sequences and Oracle Rdb not having always correctly "re−armed" the process wake−pending state by issuing a $WAKE.

This problem has been corrected in Oracle Rdb Release 7.2.3. Oracle Rdb now issues a $WAKE to the process after a hibernate stall has completed.

Application and Oracle Rdb Both Using SYS$HIBER

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

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

4.1.5 Tape Savesets With More Than 999,999 Blocks Report Wrong Block Number

Writing savesets to tape with more than 999,999 blocks reports a wrong block number using the DCL DIRECTORY command with /SIZE.

This is caused by using only 6 bytes in the ANSI HDR1 and EOF1 labels to store the block count in ASCII. A recent change in OpenVMS (BACKUP, MTAACP) extended the use of the implementation specific field in these labels to use 4 more bytes to store the high order digits for block counts larger than 999,999.

This problem has been corrected in Oracle Rdb Release 7.2.3. RMU has been changed to also use the extra 4 bytes for larger tape savesets.

4.1.6 Possible DBR Failure Due to Address Space Being Exhausted

Bug 7033216

In rare cases, it is possible for the Database Recovery (DBR) process to fail with a "footprint" similar to the following:

```
***** Exception at 00000000002EAFE0 : RDMDBR72\KUTREC$RV_CREATE + 000001E0
%COSI-F-VASFULL, virtual address space full
-SYSTEM-W-REGISFULL, specified region is full
Saved PC = 00000000002ECE60 : RDMDBR72\KUTREC$RV_GET + 000000B0
Saved PC = 00000000002EC980 : RDMDBR72\KUTREC$RV_FIND + 000004D0
Saved PC = 00000000000EFA60 : RDMDBR72\DBR$PROCESS_AIJBUF + 00001000
Saved PC = 0000000000107BC0 : RDMDBR72\DBR$REDO + 00001C50
Saved PC = 000000000010A930 : RDMDBR72\DBR$REDO_LOOP + 000001B0
Saved PC = 00000000000F8B20 : RDMDBR72\DBR$RECOVER_USER + 00001390
Saved PC = 00000000000F70D0 : RDMDBR72\DBR$RECOVER + 00000A50
```

This problem can be caused by the Database Recovery process failing to release recovery buffers in unusual circumstances. This problem may be more likely when using the row cache feature. It is also possible that other Database Recovery (DBR) process "out of memory" symptoms may be caused by the same problem.

This problem has been corrected in Oracle Rdb Release 7.2.3.

4.1.7 Control Number and Size of AIJ Initialization I/O Buffers

Bug 7040647

When an AIJ backup operation completes, the after image journal file(s) are initialized with a pattern of −1 (hex FF) bytes. This initialization is designed to be fast and thus may effectively utilize the I/O subsystem by performing large, asynchronous I/Os. This speed can, however, come at the cost of a high load on I/O components during the initialization. In rare cases, this load could slow down other I/Os on the system.

In order to allow control over the relative I/O load from the AIJ initialization operation, two logical names may be used.

- RDM$BIND_AIJ_INITIALIZE_IO_COUNT − Specifies the number of asynchronous I/O operations that will be queued at once to the AIJ file. The default value if the logical name is not defined is 2, the minimum value is 1 and the maximum value is 128.
- RDM$BIND_AIJ_INITIALIZE_IO_SIZE − Specifies the number of 512−byte disk blocks to be written per I/O. The default value if the logical name is not defined is 256 for fixed length circular journals and 32 for single extensible journals. The minimum value is 4 and the maximum value is 256.

Reducing the value of either logical might increase the amount of time needed to initialize the AIJ file after a backup. However, it may also reduce queue load on the I/O subsystem.

In prior versions of Oracle Rdb, these logical names would be translated only at database open time. They are

now also translated when processes attach to the database. This problem has been corrected in Oracle Rdb Release 7.2.3.

The values of the number of asynchronous I/O operations and number of blocks to be written per I/O for AIJ initialization may also be controlled with the RMU SHOW STATISTICS utility "AIJ Dashboard" screen using the dashboard items "Init IO Count" and "Init IO Size".

4.1.8 Some Sort Operations May Bugcheck With More Than 134 Million Rows

Bug 7113254

In some cases, a sort operation with more than 134 million rows can cause a bugcheck. For example, when building an index on a huge table, as in the following example.

```
SQL> SET TRANSACTION READ WRITE RESERVING MyTable FOR EXCLUSIVE WRITE;
SQL> CREATE INDEX MyIndex ON MyTable (
_SQL> Field1,Field2,...,Fieldn ) TYPE IS SORTED STORE IN MyIndexTable );
%RDMS-I-BUGCHKDMP, generating bugcheck dump file ...
```

The problem was caused by an erroneous check for the maximum number of rows included in a sort sequence.

This problem has been corrected in Oracle Rdb Release 7.2.3.

4.1.9 DBR Hangs on Page Lock After Node Crash

Bug 7029177

Starting with Oracle Rdb V7.1−2, under certain rare circumstances, it was possible for an Oracle Rdb Database Recovery (DBR) process to hang during database recovery after a node failure. Because the DBR process in this case was hung, all users of the database were blocked.

For this problem to occur, the database must be accessed on multiple nodes, including the one that fails. During cluster recovery, an Rdb monitor on a surviving node will create a DBR process to recover the users on the failed node. It is possible that the DBR may become blocked waiting on a page lock held by another user who, in turn, is waiting on a MEMBIT lock for the recovery to complete. Due to use of the NODLCKWT and NODLCKBLK flags on some of the locks involved, it may not be possible for the OpenVMS lock manager to detect possible deadlocks in this situation.

This problem has been corrected in Oracle Rdb Release 7.2.3.

4.1.10 Alter Storage Map ... Reorganize Pages Corrupts SORTED Indices

Bug 7022352

An ALTER STORAGE MAP ... REORGANIZE AREAS or ALTER STORAGE MAP ... REORGANIZE PAGES could cause some SORTED indices to become corrupt.

A RMU/VERIFY might show an index corruption similar to:

```
%RMU-W-BADIDXREL, Index TP_INDEX either points to a non-existent record or
                  has multiple pointers to a record in table TP.
                  The logical dbkey in the index is 3128:11:0.
%RMU-W-DATNOTIDX, Row in table TP is not in any indexes.
                  Logical dbkey is 3135:198:7.
```

A possible workaround would be to use SORTED RANKED indices.

This problem has been corrected in Oracle Rdb Release 7.2.3.

4.1.11 RDMALS Bugcheck at AIJUTL$FORMAT_ARBS + 00000DA4

Bug 6523851

On rare occasions, the AIJ Log Server (ALS) may fail with an exception at AIJUTL$FORMAT_ARBS. The bugcheck Stack Dump Summary will be similar to:

```
***** Exception at 00000000000DB154 : RDMALS72\AIJUTL$FORMAT_ARBS + 00000DA4
%COSI-F-BUGCHECK, internal consistency failure
Saved PC = 00000000000C58C8 : RDMALS72\ALS$FLUSH_ONE_CACHE + 000015C8
Saved PC = 00000000000C3C64 : RDMALS72\ALS$FLUSH + 00000664
Saved PC = 00000000000C27F4 : RDMALS72\ALS$MAIN + 00000F94
Saved PC = FFFFFFFF8013075C : symbol not found
```

The actual offset may be different depending on the platform and Rdb version being run.

This has been diagnosed to be a timing issue while the ALS is preparing updates to the After Image Journal (AIJ) file. This problem does not lead to data corruption or missing AIJ entries.

This problem has been corrected in Oracle Rdb Release 7.2.3.

4.1.12 Bugcheck Due to Command Line Too Long

Bug 7196946

The SQL Precompiler and SQL Module Language compiler support a maximum command line length of 255 characters. This length limit was not properly enforced and excessively long command lines could cause memory corruption leading to bugchecks or unexpected results.

This problem has been corrected in Oracle Rdb Release 7.2.3. The SQL Precompiler and SQL Module Language compiler now return an error message for command lines that are greater than 255 characters, as in the following example.

```
$ SQL$PRE72 -
   /CC/SQLOPT=(CONN,C_STR=NOBLANK,NOEXTERNAL_GLOBALS, -
   NODECLARE_MESSAGE_VECTOR,NOTRANSACTION)/NOMEMB -
   /NAME=SHORT/NEST=NONE/NOOPT/CHECK=NOFP_MODE/FLOAT=IEEE_FLOAT -
   /IEEE_MODE=FAST/WARNING=(DISABLE=BADBOUNDCHK)/DEBUG -
   /OBJECT=PND_XXXXX/INCL=PND_XXXXX/LIST=PND_XXXXX -
```

```
   EMPTY__MODULES_0.SC ENV_SC.SQL_CTX
%SQL-F-CMDLN_TOO_LONG, The command line is too long.
Please shorten the command
```

4.2 SQL Errors Fixed

4.2.1 ALTER PROFILE Generates a SQL Bugcheck Dump

In prior releases of Oracle Rdb, the ALTER PROFILE command would generate a SQL bugcheck when an attempt was made to change or add a DEFAULT TRANSACTION. The following example shows the problem.

```
SQL> create profile default_profile
cont> ;
SQL> alter profile default_profile
cont> default transaction read only;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=04,
virtual address=0000000000000000, PC=000000000024D484, PS=0000001B
```

To avoid this problem, use DROP PROFILE to remove the profile and recreate it. This might require using the CASCADE option and then later altering each user to replace the PROFILE.

This problem has been corrected in Oracle Rdb Release 7.2.3.

4.2.2 Incorrect File Specification Reported for RDB$STORAGE_AREAS Information Table

Bugs 6763712 and 6006502

Starting with Oracle Rdb V7.2, the domain RDB$FILE_NAME was changed to be a COMPUTED BY column that returns the file specification for the database root. For the table RDB$DATABASE, this allowed space saving in the root file. Unfortunately, this domain was inadvertently used by several information tables. This lead to unexpected results for those file specifications from these tables.

The following example shows that when selecting the names of the snapshot files from the table RDB$STORAGE_AREAS, only the root file specification is returned.

```
SQL> SELECT RDB$AREA_FILE AS SNAP_NAME EDIT USING 'T(50)',
cont> RDB$INITIAL_ALLOCATION AS INITIAL EDIT USING 'Z(9)',
cont> RDB$CURRENT_ALLOCATION AS CURRENT EDIT USING 'Z(9)'
cont> FROM RDB$STORAGE_AREAS
cont> WHERE (RDB$CURRENT_ALLOCATION = RDB$INITIAL_ALLOCATION) AND
cont> (RDB$AREA_NAME = ' ') limit to 3 rows;
SNAP_NAME INITIAL CURRENT
DISK14:[TESTING.DATABASE]SAMPLE.RDB;1 5000 5000
DISK14:[TESTING.DATABASE]SAMPLE.RDB;1 500 500
DISK14:[TESTING.DATABASE]SAMPLE.RDB;1 1000 1000
3 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.2.3. The INFO_TABLES.SQL script now uses a new domain RDB$FILE_SPECIFICATION. If these information tables exist in a V7.2 database, they must be dropped and replaced using the revised SQL$SAMPLE:INFO_TABLES.SQL script.

Example

This example shows the execution of the SQL$SAMPLE:INFO_TABLES.SQL script to replace the information tables.

```
SQL> attach 'filename MF_PERSONNEL';
SQL> @info_tables

 Copyright (c) 1997, 2008, Oracle Corporation.  All Rights Reserved.

 Please ignore any FIELD_EXISTS error for the domain RDB$FILE_SPECIFICATION
%SQL-F-FIELD_EXISTS, Domain RDB$FILE_SPECIFICATION already exists in this
database or schema

 Creating RDB$CACHE

 Creating RDB$DATABASE_JOURNAL

 Creating RDB$DATABASE_ROOT

 Creating RDB$DATABASE_USERS
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-VIEWEXI, relation RDB$DATABASE_USERS is referenced by view V$X
-RDMS-F-RELNOTDEL, relation RDB$DATABASE_USERS has not been deleted
%SQL-F-REL_EXISTS, Table RDB$DATABASE_USERS already exists in this database or
schema

 Creating RDB$STORAGE_AREAS

 Creating RDB$JOURNALS

 Creating RDB$LOGICAL_AREAS

 Creating RDB$CHARACTER_SETS

 Creating RDB$NLS_CHARACTER_SETS

 Type COMMIT if there were no unexpected errors, otherwise ROLLBACK
SQL> rollback;
```

Usage Notes

- The INFO_TABLES script attempts to define a special domain RDB$FILE_SPECIFICATION. If this domain already exists in the database, then it is reasonable to see this message and it can be ignored.
- If one of the existing information tables is referenced by a view or stored procedure, then SQL will report the dependency on that object. Assuming that the sources for that view exist for recreation in the database, the database administrator can manually drop the table (RDB$DATABASE_USERS in this example) using the CASCADE option. Once INFO_TABLES.SQL is re−executed and committed then the views can be recreated. Stored procedures are not dropped by the CASCADE option so will execute correctly when the information table is recreated.

4.2.3 Unexpected Bugcheck When Processing Zero Length Strings

Bug 6834122

In prior versions of Oracle Rdb, generation of zero length string literals might cause a bugcheck, as shown in the following example.

```
SQL> create table test_char(fld1 char(10));
SQL> insert into test_char values (''||'');
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER1:[TESTING]SQLBUGCHK.DMP;
%SQL-F-BUGCHK, There has been a fatal error. Please contact your Oracle
support representative. SQL$BLRXPR - 30
```

This problem has been corrected in Oracle Rdb Release 7.2.3. Oracle Rdb now correctly handles this case. As a workaround, this query will succeed if the dialect is changed to either ORACLE LEVEL1 or ORACLE LEVEL2.

4.2.4 Unexpected Bugcheck from RENAME TABLE Statement

In prior releases of Oracle Rdb, the RENAME TABLE command would generate a bugcheck if the target table was a GLOBAL or LOCAL TEMPORARY table. The command was attempting to modify the Rdb AIP structure which is not used by temporary tables.

The following shows an example of this problem.

```
SQL> create global temporary table SAMPLE
cont>   (a integer)
cont>   on commit preserve rows;
SQL> rename table SAMPLE to SAMPLING;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]RDSBUGCHK.DMP;
```

This problem has been corrected in Oracle Rdb Release 7.2.3.

4.2.5 Unexpected RDMS-F-NODBK Reported by DROP USER Statement

Bug 6959409

In prior releases of Oracle Rdb, the DROP USER statement could report a RDMS-F-NODBK when processing a database defined with SECURITY CHECKING IS INTERNAL. In this type of database, the DROP USER also implicitly performs REVOKE ENTRY on each access control list (ACL) defined for all objects in the database. During this action, it is possible that the database key for the access control list will not be saved correctly.

The following example shows the type of error reported.

```
SQL> drop user rdbtest restrict;
%RDB-E-NO_RECORD, access by dbkey failed because dbkey is no longer associated
with a record
-RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-NODBK, 0:67108864:0 does not point to a data record
SQL>
```

It is possible that defining RDM$BIND_BUFFERS to a sufficiently large value may avoid this problem. This

problem has been corrected in Oracle Rdb Release 7.2.3. Oracle Rdb has been corrected to save the reference to the LIST OF BYTE VARYING access control list.

4.2.6 Unexpected COSI−F−VASFULL Reported by Long Running SQL Application

Bug 6952372

In some rare cases, SQL precompiler or SQL Module Language applications may leak memory when fetching numeric values into host variables defined as character types (for instance char in C, or PIC X in COBOL).

The following example shows an extreme case where a long running application aborted because of insufficient virtual address space.

```
$ R RDB_V_TEST3
%RDB−E−ARITH_EXCEPT, truncation of a numeric value at runtime
−RDB−E−ARITH_EXCEPT, truncation of a numeric value at runtime
−COSI−F−VASFULL, virtual address space full
−SYSTEM−F−ILLPAGCNT, illegal page count parameter
%SYSTEM−E−ABORT, abort
$ exit
```

This problem has been corrected in Oracle Rdb Release 7.2.3. This problem was caused by an incorrectly initialized descriptor that contained an invalid character set identification.

Oracle recommends that any affected application be rebuilt once this release is installed. This rebuild will require recompile with either the SQL precompiler or the SQL module language compilers.

In addition to this correction, some performance work was implemented in this area that will benefit all SQL precompiler or SQL Module Language applications.

4.2.7 Diagnostics Lost Following Procedure Call in Dynamic SQL

Bug 6379149

In prior versions of Oracle Rdb, it was possible that an exception raised by a stored procedure or a multi−statement procedure could be lost when executed from Dynamic SQL. The procedure must issue a COMMIT or ROLLBACK prior to raising the exception.

The following example shows that the signalled error (in this case from an explicit SIGNAL statement) is not reported through Dynamic SQL. The reported case was missing an exception reporting a failed constraint validation. Note: the output in this example is from a Dynamic SQL testing tool.

```
−> ATTACH 'filename sql$database';
−> SET DIALECT 'oracle level2';
−> SET HOLD CURSORS 'all';
−> SET TRANSACTION READ WRITE;
−> ! OCI ALWAYS DECLARES CURSORS AS WITH HOLD PRESERVE ALL, SEE SET HOLD
CURSORS ABOVE;
−> DYN P SELECT RDB$RELATION_NAME FROM RDB$RELATIONS;
```

```
Embedded DYN PREP executed (Implicit).
-> DYN O;
Embedded DYN OPEN executed (Implicit).
-> DYN F;
 0/RDB$RELATION_NAME: CANDIDATES
Embedded DYN FETCH executed (Implicit).
-> \
Entering block mode...
-> BEGIN
-> COMMIT;
-> SIGNAL 'RR002' ('unexpected error');
-> END;
-> \
Leaving block mode...
-> COMMIT;
-> EXIT;
```

For this erroneous behavior to be observed, the following must be true:

- A cursor must be open at the time that the procedure (stored or multi−statement) is executed.
- The procedure must execute a COMMIT or ROLLBACK statement.
- The cursor must be a WITH HOLD cursor otherwise the COMMIT or ROLLBACK will implicitly close any open cursors. This can be achieved using the WITH HOLD clause on DECLARE CURSOR or by executing the command SET HOLD CURSORS 'all'. Cursors open though OCI Services for Rdb are implicitly WITH HOLD cursors.

The returned diagnostics were lost when SQL checked that all non−HOLD cursors were closed following a transaction state change made by the procedure call. This problem has been corrected in Oracle Rdb Release 7.2.3. The returned diagnostics are now preserved and reported.

4.2.8 Unexpected Bugcheck from SET TRANSACTION Statement

Bug 7010980

In rare cases, Oracle Rdb would generate a bugcheck dump during a SET TRANSACTION statement that included a PARTITION list as part of the RESERVING clause.

The following example shows this problem.

```
SQL> set transaction
cont>     read only
cont>     wait 0
cont>     reserving
cont>         table1 for shared read,
cont>         table2 for shared read,
cont>         table3 for shared read,
cont>         table70 for shared read,
cont>         table83 for shared read,
cont>         table101 for shared read,
cont>         table367 partition(15) for shared read;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=000000000225E7DC, PC=00000000001F48DC, PS=0000001B
```

This problem only occurs when the database contains many tables, indices and partitions and one of the tables referenced in the SET TRANSACTION statement has a logical area much larger than that referenced in the PARTITION clause.

This problem has been corrected in Oracle Rdb Release 7.2.3.

4.3 RMU Errors Fixed

4.3.1 RMU Extract Reports BAD_CODE Error for TRANSLATE Function

Bug 6752325

In prior releases of Oracle Rdb, RMU Extract would generate a BAD_CODE error when trying to extract the TRANSLATE function. The following example shows the reported error.

```
%RMU-F-BLRINV, internal error - BLR string 83 for SAMPLE_VIEW. is invalid
-COSI-E-BAD_CODE, corruption in the query string
%RMU-F-FTL_RMU, Fatal error for RMU operation at 16-JAN-2008 06:52:12.81
```

This problem has been corrected in Oracle Rdb Release 7.2.3.

4.3.2 Restore of Selected Area Fails With ACCVIO

A restore using a journal file to restore one or more areas from a multi-volume or multi-file backup fails with an ACCVIO.

```
$ RMU/RESTORE/AREA/JOURNAL=JF/REWIND/VOLUME=3 -
      TAPE1:MFP/MASTER,TAPE2:/MASTER,TAPE3:/MASTER DEPARTMENTS
...
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
      virtual address=0000000048, PC=...
```

Using the journal file, RMU realized the storage area DEPARTMENTS was located on the first tape volume. But remaining code still tried to access the other tapes without a proper parameter block.

This problem has been corrected in Oracle Rdb Release 7.2.3. The code now does not try to access an input volume or input file that is not needed.

4.3.3 RMU Load Quietly Truncated String Data During Insert

Bug 6732438

In prior versions of Oracle Rdb, the RMU Load command would quietly truncate data loaded into CHAR and VARCHAR columns. This loss of data might be significant but was never reported by Oracle Rdb.

This problem has been corrected in Oracle Rdb Release 7.2.3. RMU Load now defaults to SQL dialect SQL99 which implicitly checks for and reports truncations during INSERT operations.

This behavior is controlled by a new /DIALECT qualifier.

- /NODIALECT, /DIALECT=SQL89 or /DIALECT=NONE will revert to prior behavior and no truncation error will be reported.

- /DIALECT=SQL99 (the default) will enable reporting of truncation errors. Note that truncation occurs if non−space characters are discarded during the insert.

```
$ rmu/load abc f2 f1
%RMU−I−LOADERR, Error loading row 1.
%RDB−E−TRUN_STORE, string truncated during assignment to a column
%RMU−I−DATRECREAD,  1 data records read from input file.
%RMU−I−DATRECSTO,   0 data records stored.
%RMU−F−FTL_LOAD, Fatal error for LOAD operation at 13−FEB−2008 15:39:44.40
$
```

4.3.4 RMU /UNLOAD /AFTER_JOURNAL Sort Performance

The RMU /UNLOAD /AFTER_JOURNAL command sorts all records for each transaction in order to remove duplicate modifications to the same record within a transaction. Previous versions of the RMU /UNLOAD /AFTER_JOURNAL command used the "SORT32" package and an internal sort algorithm to perform this sorting. However, for certain numbers of records being sorted, the internal sort algorithm could consume an unexpectedly large amount of CPU time.

The RMU /UNLOAD /AFTER_JOURNAL command has been improved with a significantly faster internal sort algorithm. This should result in significant performance improvements for certain cases of extracting data from the after image journal file.

4.3.5 Possible RMU/MOVE_AREA/ONLINE Database Corruption

Bug 6664950

RMU/MOVE/ONLINE could fail to correctly move Oracle Rdb database storage areas if other users were modifying the same areas. The Move operation could complete without any errors. Or, the Move operation would fail to delete the original area file(s) that had been moved because another database user incorrectly still had access to them.

```
%RMU−F−CANTDELETE, error deleting "DISK:[DIRECTORY]AREA.RDA;1"
%COSI−E−FLK, file currently locked by another user
−RMS−E−FLK, file currently locked by another user
%RMU−F−CANTDELETE, error deleting "DISK:[DIRECTORY]AREA.SNP;1"
%COSI−E−FLK, file currently locked by another user
−RMS−E−FLK, file currently locked by another user
```

An RMU/VERIFY of the database done after the move would show one of the following errors:

The first type of verify error reports that rows have been inserted in the storage area moved that were not in an index.

```
RMU/VERIFY/ALL DATABASE.RDB
%RMU−W−DATNOTIDX, Row in table ACCOUNT is not in any indexes.
                  Logical dbkey is  68:270:2.
%RMU−W−DATNOTIDX, Row in table ACCOUNT is not in any indexes.
                  Logical dbkey is 68:343:1.
```

The second type of verify error reports that the maximum page number in the database root for the storage area snapshot file moved is invalid.

```
RMU/VERIFY/ALL DATABASE.RDB
%RMU-E-BADMAXPNO, unable to read last page (1478) in file
DISK:[DIRECTORY.ONE]AREA.SNP;1
%RMU-F-ABORTVER,  fatal error encountered; aborting verification
```

If a previous version of Rdb is installed which does not include the solution to this problem, it is highly recommended not to use online moves while the storage area being moved is being modified.

The following example of this problem shows an online move of the area HASHES of the database MOVIE from DISK:[DIRECTORY] to DISK:[DIRECTORY.ONE]. At the end of the move, the %RMU−F−CANTDELETE error was output when RMU attempted to delete the original source storage area files that had been moved. This occurred because another user still had access to these files because of this problem.

```
$ rmu/move/online/log/directory=[.one] movie.rdb hashes
%RMU-I-QUIETPT, waiting for database quiet point at 26-NOV-2007
10:44:23.86
%RMU-I-RELQUIETPT,  Database quiet point lock has been released at
26-NOV-2007 10:44:33.39.
%RMU-I-MOVTXT_04, Starting move of storage area
DISK:[DIRECTORY.ONE]HASHES.RDA;1 at 26-NOV-2007 10:44:35.68
%RMU-I-MOVTXT_01, Completed move of storage area
DISK:[DIRECTORY.ONE]HASHES.RDA;1 at 26-NOV-2007 10:44:40.44
%RMU-I-MOVTXT_05, Moved snapshot area file
DISK:[DIRECTORY.ONE]HASHES.SNP;1
%RMU-I-MOVTXT_15, Live area files moved
%RMU-I-MOVTXT_10, Root entry modified for moved area file
DISK:[DIRECTORY.ONE]HASHES.RDA;1
%RMU-I-MOVTXT_11, Root entry modified for moved snapshot file
DISK:[DIRECTORY.ONE]HASHES.SNP;1
%RMU-I-MOVTXT_06, Database root updated for all moved areas
%RMU-I-MOVTXT_16, On line users notified of completion of move
%RMU-F-CANTDELETE, error deleting "DISK:[DIRECTORY]HASHES.RDA;1"
%COSI-E-FLK, file currently locked by another user
-RMS-E-FLK, file currently locked by another user
%RMU-F-CANTDELETE, error deleting "DISK:[DIRECTORY]HASHES.SNP;1"
%COSI-E-FLK, file currently locked by another user
-RMS-E-FLK, file currently locked by another user
%RMU-I-MOVTXT_07, Obsolete files deleted
%RMU-W-DOFULLBCK, full database backup should be done to ensure future
recovery
%RMU-I-COMPLETED, MOVE_AREA operation completed at 26-NOV-2007
10:44:41.80
```

A workaround to avoid this problem is to not specify /ONLINE for the move command. If /ONLINE is specified, always verify the database after the move. If the verify shows the index diagnostics, the indexes should be rebuilt. If the verify shows the root diagnostics, the database can be restored and the move repeated without specifying /ONLINE.

This problem has been corrected in Oracle Rdb Release 7.2.3.

4.3.6 RMU /BACKUP Performance Enhanced On Some Types Of Tape Drives

Performance testing of the RMU/BACKUP utility using LTO4 tape drives indicated that some magnetic tape devices experienced unexpectedly long delays while repositioning after writing tape marks. This delay was found to have been the result of not using the "fast" method of skipping over tape marks. In some situations, this delay was found to be several minutes long.

This problem has been corrected in Oracle Rdb Release 7.2.3. Performance of the RMU/BACKUP utility has been improved for magnetic tape devices that support the SCSI space−by−file−marks command and the SCSI read position command.

As a workaround, the DCL command "SET MAGTAPE/FAST_SKIP=ALWAYS dev" can be used to set the tape drive to use the "fast" method of skipping over tape marks by default.

4.3.7 RMU Analyze Reports Incorrect Compression Ratio for Some Tables

Bug 6992051

Starting with Oracle Rdb V7.1, the run−length compression algorithm was changed such that very long strings are compressed using a more compact encoding scheme. Unfortunately, the RMU/ANALYZE command did not use the new algorithm and so in some cases miscalculated the compression ratio for some table logical areas.

The following example shows a fragment of output from the RMU Analyze command.

```
Logical area: COL1 for storage area : BIGAREA
Larea id: 57,  Record type: 30, Record length: 3608, Compressed

Data records: 1, bytes used: 68 (0%)
  average length: 68, compression ratio: 0.45
```

In this example, the stored rows were 3603 bytes long (record length minus the row overhead) and therefore the compression ratio should have been about 0.02.

This problem has been corrected in Oracle Rdb Release 7.2.3. RMU Analyze now uses the correct algorithm to compute the expanded row length and to compute a more accurate compression ratio.

4.3.8 Restore May Miss Storage Area Without Warning

A restore of a full multi−volume or multi−file backup may complete without a warning even though not all savesets have been specified in the restore command. Hence storage areas saved in the missing saveset(s) are not restored.

A RMU/VERIFY of the restored database does not always detect the error.

```
$ RMU/BACKUP/DISK=WRITER_THREADS:2 MF_PERSONNEL MF_PERSONNEL.RBF,[]
$ SQL DROP DATABASE FILE MF_PERSONNEL;
```

```
$ !
$ ! Restore command with missing 2nd save set
$ !
$ RMU/RESTORE/NOCDD/DISK MF_PERSONNEL.RBF
%RMU-I-AIJRSTAVL, 0 after-image journals available for use
%RMU-I-AIJISOFF, after-image journaling has been disabled
%RMU-W-USERECCOM, Use the RMU Recover command. The journals are not available.
$ RMU/VERIFY/ALL/NOLOG MF_PERSONNEL
%RMU-W-BADPROID,  DKA100:[TEST]EMPIDS_LOW.RDA;1 file contains a bad identifier
Expected "RDMSDATA", found ""
%RMU-W-BADDBPRO,  DKA100:[TEST]EMPIDS_LOW.RDA;1 file does not belong to
                  database DKA100:[TEST]MF_PERSONNEL.RDB;1
                  found references to database
...
$ !
$ ! Correct restore command
$ !
$ RMU/BACKUP/DISK=READER_THREADS:2 MF_PERSONNEL MF_PERSONNEL.RBF,[]
```

A check has been added to RMU/RESTORE which issues a warning message after the completion of the restore if not all areas have been restored.

```
%RMU-W-RESINCOMP, Not all storage areas have been restored

Explanation:
        One or more storage areas could not be restored.

User Action:
        Use RMU/VERIFY to find any missing or incomplete
        storage area. Repeat the restore command and include
        the save sets with the missing storage areas.
```

This problem has been corrected in Oracle Rdb Release 7.2.3.

4.3.9 Incorrect View Syntax Generated by RMU Extract Item=VIEW

Bug 6905847

Prior releases of Oracle Rdb would not correctly extract views generated by third-party database tools. The following example shows the incorrect syntax in the SELECT value expression list.

```
$ rmu/extract-
    /item=view-
    /option=(noheader,mat:V2%,filename_only) abc
set verify;
set language ENGLISH;
set default date format 'SQL92';
set quoting rules 'SQL92';
set date format DATE 001, TIME 001;
attach 'filename ABC';
create view "V2"
    (A,
     B,
     C,
     D,
     E) as
    (select
```

```
    ,
    ,
        C1.C,
        C1.D,
        C1.E
    from ABCD C1);

commit work;
```

This problem has been corrected in Oracle Rdb Release 7.2.3. RMU Extract now handles the variations in these view definitions and generates a valid SQL view definition.

4.3.10 Revised Format for the RMU/UNLOAD/AFTER_IMAGE CONTROL File

In prior releases of Oracle Rdb, the command RMU/UNLOAD/AFTER_IMAGE permitted the unloaded after−image table data to be loaded into an Oracle RDBMS database using the SQL*Loader tool. This was enabled by generating a control file for SQL*Loader using the CONTROL qualifier and outputting the data in a portable TEXT format. There were several problems with this support that have been been corrected in this release.

- The LogMiner timestamp information was described as Oracle DATE type which does not support fractional seconds precision. Now the fields RDB$LM_START_TAD and RDB$LM_COMMIT_TAD are defined as TIMESTAMP types with fractional seconds precision.
- The syntax generated for the control file was incorrect in several places. This included specifying 12 hour date/time masks instead of 24 hour masks for DATE, incorrect type specifiers for numeric values, and incorrect offset values.
- For some data types, the unloaded TEXT format was too narrow allowing possible truncation of the data and for some too wide, wasting space in the unloaded data file. This has been corrected for both LogMiner and RMU/UNLOAD for FORMAT=TEXT.

This problem has been corrected in Oracle Rdb Release 7.2.3. At this time only partial support for TIME and INTERVAL is available in the RMU/UNLOAD/AFTER_IMAGE CONTROL file.

4.3.11 RMU Created an Incorrect Restore Options File for Single File Databases

Bug 7033861

The Oracle Rdb RMU commands RMU/DUMP/RESTORE_OPTIONS, RMU/DUMP/BACKUP/RESTORE_OPTIONS and RMU/BACKUP/RESTORE_OPTIONS created an incorrect options file to be used by the RMU/RESTORE/OPTIONS command. The file specification of the database in the first comment line was missing. The "/FILE" qualifier, illegal for single file databases which do not have an *.RDA storage area file, was included and was not followed by a file name if the database had not yet been backed up. The /SNAPSHOT qualifier "FILE=" parameter was also included when it should not have been and was also not followed by a file specification if the database had not yet been backed up. The /SNAPSHOT qualifier "FILE=" parameter in an options file for a single file database cannot be used to change either the location or name of the snapshot file.

For an Oracle Rdb single file database, the single storage area is located in the *.RDB file along with the database root. There is only one snapshot file which has the same name as the database *.RDB file. For example, for the single file personnel database, the single storage area is located in the PERSONNEL.RDB file and the single snapshot file is named PERSONNEL.SNP. The option file used by the restore command can be used to change other parameters of the single storage area but not the *.RDB and *.SNP file names and locations.

The following example of this problem shows that the REST.OPT options file created by the RMU/DUMP command had incorrect syntax which caused an error when the RMU/RESTORE/OPTION=REST.OPT command was executed.

```
$ RMU/DUMP/OUT=NL:/RESTORE_OPT=REST.OPT PERSONNEL.RDB
$ TYPE REST.OPT
!  Options file for database
!  Created dd-mmm-yyyy hh:mm:ss.xxxx
! Created by DUMP command

RDB$SYSTEM -
        /file= -
        /extension=ENABLED -
        /read_write -
        /spams -
        /snapshot=(allocation=100, -
                   file=)
$ RMU/RESTORE/NOLOG/OPTION=REST.OPT/NOCDD PERS.RBF
%RMU-F-CONFLSWIT, conflicting qualifiers /FILE and single file
 database organization
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at dd-mmm-yyyy hh:mm:ss.xxxx
```

A workaround to avoid this problem is to edit the options file to eliminate the "/file=" qualifier and the "file=" parameter that follows the "/snapshot" qualifier and add the database file specification to the end of the first comment line. Here is the correct restore options file format for a single file database that the RMU commands named above will output now that this problem has been corrected.

```
$ RMU/DUMP/OUT=NL:/RESTORE_OPT=REST.OPT PESONNEL.RDB
$ TYPE REST.OPT
!  Options file for database SERDB_USER1:[HOCHULI.BUG7033861]PERSONNEL.RDB;1
!  Created 25-JUN-2008 14:51:31.90
!  Created by DUMP command

RDB$SYSTEM -
        /extension=ENABLED -
        /read_write -
        /spams -
        /snapshot=(allocation=100)
$ RMU/RESTORE/NOLOG/OPTION=REST.OPT/NOCDD PERS.RBF
%RMU-I-AIJRSTAVL, 0 after-image journals available for use
%RMU-I-AIJISOFF, after-image journaling has been disabled
%RMU-W-USERECCOM, Use the RMU Recover command. The journals are not
 available.
```

This problem has been corrected in Oracle Rdb Release 7.2.3.

4.4 RMU Show Statistics Errors Fixed

4.4.1 RMU/SHOW STATISTICS Checkpoint Information Screen Sort By Oldest Transaction Did Not Work Correctly

Bug 6735923

In previous Rdb 7.2 releases, the RMU /SHOW STATISTICS "Checkpoint Information" screen option to "Sort by oldest transaction" option did not work properly. Users were unable to order the display by transaction start time.

This problem has been corrected in Oracle Rdb Release 7.2.3. The "Sort by oldest transaction" option now works as expected.

4.4.2 Date and Time Included In Lock Deadlock Log File

Bug 4069186

Previously, only the time of a deadlock was included in the "Lock Deadlock Log" file. In some cases, the lack of the date field made analysis difficult.

This problem has been corrected in Oracle Rdb Release 7.2.3. The "Lock Deadlock Log" file now always includes a date field as demonstrated in this example output.

```
 Oracle Rdb V7.2−200 Performance Monitor Lock Deadlock Log
 Database DPA10:[DB]AARCH.RDB;1
 Lock Deadlock Log created 20−MAR−2008 12:18:41.18
2040045C:1  20−MAR−2008 18:11:20.3145724 − waiting for page 18:8 (PW) [6 missed]
20400DA3:1  21−MAR−2008 08:00:01.9816693 − waiting for page 53:17111 (PW)
```

4.4.3 Incorrect RMU/SHOW STATISTICS Layout File RMU$SHOW_STATISTICS72.CDO

Bug 7156903

The file RMU$SHOW_STATISTICS72.CDO that shipped with prior versions of Oracle Rdb V7.2 did not correctly describe the layout of the RMU/SHOW STATISTICS data file.

This problem has been corrected in Oracle Rdb Release 7.2.3. A revised RMU$SHOW_STATISTICS72.CDO file is provided by this release. In addition two companion files are provided in SYS$LIBRARY: RMU$SHOW_STATISTICS72.SQL, which creates a SQL table for use by RMU/LOAD; and RMU$SHOW_STATISTICS72.H, a C header file to simplify processing of the data file by C applications.

Chapter 5
Enhancements And Changes Provided in Oracle Rdb Release 7.2.3.2

5.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.3.2

5.1.1 New SQL Functions Added

Bug 7365167

This release of Oracle Rdb adds new functions to the SYS$LIBRARY:SQL_FUNCTIONS72.SQL script. To replace the existing library of functions first use the SQL_FUNCTIONS_DROP72.SQL script and then reapply using SQL_FUNCTIONS72.SQL.

Description

- ACOS returns the arc cosine of n. The argument n must be in the range of −1 to 1, and the function returns a DOUBLE PRECISION value in the range of 0 to pi, expressed in radians.
  If the passed expression results in NULL, then the result of ACOS will be NULL.
  The following example returns the arc cosine of .3:

```
SQL> SELECT ACOS(.3) "Arc_Cosine" FROM Rdb$DATABASE;
              Arc_Cosine
  1.266103672779499E+000
1 row selected
```

- ACOSH returns the hyperbolic arc cosine of n. The argument n must be equal to or greater than 1. The function returns a DOUBLE PRECISION value.
  If either passed expression results in NULL, then the result of ACOSH will be NULL.

```
SQL> SELECT ACOSH(1.0) "Hyperbolic Arc Cosine" FROM Rdb$DATABASE;
   Hyperbolic Arc Cosine
  0.000000000000000E+000
1 row selected
```

- ASIN returns the arc sine of n. The argument n must be in the range of −1 to 1, and the function returns a DOUBLE PRECISION value in the range of −pi/2 to pi/2, expressed in radians.
  If the passed expression results in NULL, then the result of ASIN will be NULL.
  The following example returns the arc sine of .3:

```
SQL> SELECT ASIN(.3) "Arc_Sine" FROM Rdb$DATABASE;
                Arc_Sine
  3.046926540153975E-001
1 row selected
```

- ASINH returns the hyperbolic arc sine of n. The function returns a DOUBLE PRECISION value.
  If either passed expression results in NULL, then the result of ASINH will be NULL.

```
SQL> SELECT ASINH (-90.0) "Hyperbolic Arc Sine" FROM Rdb$DATABASE;
     Hyperbolic Arc Sine
 -5.192987713658941E+000
1 row selected
```

- ATAN returns the arc tangent of n. The argument n can be in an unbounded range and returns a value in the range of −pi/2 to pi/2, expressed in radians.
  If the passed expression results in NULL, then the result of ATAN will be NULL.
  The following example returns the arc tangent of .3:

```
SQL> SELECT ATAN(.3) "Arc_Tangent" FROM Rdb$DATABASE;
             Arc_Tangent
  2.914567944778671E-001
1 row selected
```

- ATAN2 returns the arc tangent of n and m. The argument n can be in an unbounded range and returns a value in the range of −pi to pi, depending on the signs of n and m, expressed in radians.
  ATAN2(n,m) is the same as ATAN(n/m).
  If either passed expression results in NULL, then the result of ATAN2 will be NULL.

```
SQL> SELECT ATAN2(.3, .2) "Arc_Tangent2" FROM Rdb$DATABASE;
            Arc_Tangent2
  9.827937232473291E-001
1 row selected
```

- ATANH returns the hyperbolic arc tangent of n (in radians). The argument n must be in the range of −1 to 1, and the function returns a DOUBLE PRECISION value.
  If either passed expression results in NULL, then the result of ATANH will be NULL.

```
SQL> SELECT ATANH(0.905148254) "Hyperbolic Arc Tangent" FROM Rdb$DATABASE;
  Hyperbolic Arc Tangent
  1.500000001965249E+000
1 row selected
```

- BITAND computes an AND operation on the bits of expr1 and expr2, both of which must resolve to integers, and returns an integer. This function is commonly used with the DECODE function, as illustrated in the example that follows.
  If the passed expression results in NULL, then the result of BITAND will be NULL.
  The first bit of the mask stored in the column RDB$FLAGS of the table Rdb$RELATIONS indicates that this relation is a view definition. This query displays the names of any views in the database.

Example 5−1 Checking bits in RDB$FLAGS column

```
SQL> -- Which objects in Rdb$RELATIONS are views?
SQL> select rdb$relation_name from rdb$relations where bitand(rdb$flags, 1) = 1;
 RDB$RELATION_NAME
 RDBVMS$COLLATIONS
 RDBVMS$INTERRELATIONS
 RDBVMS$PRIVILEGES
 RDBVMS$RELATION_CONSTRAINTS
 RDBVMS$RELATION_CONSTRAINT_FLDS
 RDBVMS$STORAGE_MAPS
 RDBVMS$STORAGE_MAP_AREAS
 RDBVMS$TRIGGERS
8 rows selected
```

This example uses the result of the BITAND in a DECODE list to display attributes of an Rdb database.

Example 5−2 Using BITAND with DECODE

```
SQL> select
cont>     DECODE (BITAND (rdb$flags,1), 0, 'No Dictionary', 'Dictionary'),
cont>     DECODE (BITAND (rdb$flags,2), 0, 'ACL Style', 'ANSI Style'),
cont>     DECODE (BITAND (rdb$flags,64), 0, 'No Multischema', 'Multischema')
cont> from
cont>     Rdb$DATABASE;
```

```
 No Dictionary   ACL Style    Multischema
1 row selected
SQL>
```

- COT returns the cotangent of n. The function returns a DOUBLE PRECISION value. If either passed expression results in NULL, then the result of COT will be NULL.

```
SQL> SELECT COT (3.14159265358979/4) "Cotangent" FROM Rdb$DATABASE;
               Cotangent
  1.000000000000002E+000
1 row selected
```

5.1.2 RMU /SHOW STATISTICS Enhanced LogMiner Information Display

The RMU /SHOW STATISTICS "LogMiner Information" display has been enhanced to include additional information about the state of Continuous LogMiner processes.

Information displayed includes the following fields (note that the terminal screen width must be greater than 80 columns to display all fields):

- Process.ID − The Process and Stream ID of the Continuous LogMiner process
- State − The current Continuous LogMiner process state:
  - ♦ Inactive
  - ♦ Hibernating
  - ♦ Polling
  - ♦ Extracting
- SeqNum − The AIJ sequence number currently being read
- CurrVBN − The disk block number currently being processed
- Actv − The number of active transactions being processed
- MQP_VNO − The AIJ sequence number location of the last micro−quiet point detected
- MQP_VBN − The disk block number location of the last micro−quiet point detected
- MQP_TSN − The transaction sequence number of the last micro−quiet point detected
- LastTSN − The transaction sequence number of the last transaction

5.1.3 RMU /SHOW STATISTICS "Checkpoint Statistics" New Counters

Previously, on the "Checkpoint Statistics" screen of the "RMU /SHOW STATISTICS" utility, two of the possible checkpoint conditions were not being captured for display. These conditions now displayed are:

- "Clear" indicates that a process's checkpoint information is to be cleared in the root file.
- "Initial" indicates that a process has performed its initial checkpoint to establish a checkpoint starting location.

Aggregate Total

Because some of the checkpoint conditions can occur in combination, the sum total of all of the possible checkpoint types may exceed the aggregate total "checkpoints" value displayed.

5.1.4 SQL Enhancements: Allowing Optional Correlation Names

Bug 6847694

This release of Oracle SQL improves compatibility with Oracle RDBMS SQL. In particular, some SQL syntax sent via OCI Services for Rdb was rejected in prior releases.

Rdb SQL contains the following enhancements:

1. Oracle RDBMS does not require the correlation name for a derived table (which can be deeply nested). However, in prior releases, Oracle RDB would reject this syntax.

   ```
   SQL> select *
   cont> from (select last_name, first_name from candidates);
   %SQL-F-CORNAMREQ, A correlation name is required for a derived table
   SQL>
   SQL> select *
   cont> from (select last_name, first_name
   cont>       from (select last_name, first_name from candidates));
   %SQL-F-CORNAMREQ, A correlation name is required for a derived table
   ```

   Oracle Rdb now also supports an optional correlation name.

   ```
   SQL> select *
   cont> from (select last_name, first_name from candidates);
    LAST_NAME        FIRST_NAME
    Wilson           Oscar
    Schwartz         Trixie
    Boswick          Fred
   3 rows selected
   SQL>
   SQL> select *
   cont> from (select last_name, first_name
   cont>       from (select last_name, first_name from candidates));
    LAST_NAME        FIRST_NAME
    Wilson           Oscar
    Schwartz         Trixie
    Boswick          Fred
   3 rows selected
   ```
2. Oracle RDBMS allows an optional correlation name for INSERT targets. Oracle Rdb did not permit a correlation name in this location.

   ```
   SQL> create table fred (a integer);
   SQL> insert into fred f (a) values (1);
   insert into fred f (a) values (1);
                          ^
   %SQL-W-LOOK_FOR_STT, Syntax error, looking for:
   %SQL-W-LOOK_FOR_CON,            AS, RETURNING, PLACEMENT, ;,
   ```

```
%SQL-F-LOOK_FOR_FIN,    found VALUES instead
```

Oracle Rdb now also supports an optional correlation name.

```
SQL> create table fred (a integer);
SQL> insert into fred f (a) values (1);
1 row inserted
```

The correlation name is useful for the RETURNING clause.

```
SQL> insert into fred f (a) values (2) returning f.a;
           A
           2
1 row inserted
```

These problems have been corrected in Oracle Rdb Release 7.2.3.2.

5.1.5 Performance Enhancements With Internal Lock Data Structures

In order to help performance for certain classes of applications that lock a large number of records within a transaction, several optimizations have been implemented:

- An internal hash table used to access a list of record locks owned by a database user has been increased in size to help speed access to entries within the table.
- An internal data structure used in conjunction with the hash table is now allocated in larger segments to reduce the number of memory allocations.

5.1.6 Change in Frequency of Cardinality Updates Might be Observed

With this release, Oracle Rdb changed slightly the frequency at which table and index cardinality changes, or workload column group rows (RDB$WORKLOAD) were flushed to the system tables. Some users of RMU/UNLOAD/AFTER_IMAGE or RMU/OPTIMIZE/AFTER_IMAGE may observe one more or one less transaction being processed than in prior releases. The reason for this difference is that Rdb attempts to update the system tables on the tail end of a user's READ WRITE transaction and the change in frequency may or may not require Rdb to start an additional transaction during the DISCONNECT processing in which to write back the final statistics.

5.1.7 New Interactive SQL Statements

This release of Oracle Rdb adds new Interactive SQL statements to improve compatibility with SQL*plus.

- SET FEEDBACK { ON | OFF n }
  The existing SET FEEDBACK statement now supports a numeric option that defines the threshold at which reported line counters are displayed.

```
SQL> set feedback 2
SQL> select * from work_status;
 STATUS_CODE   STATUS_NAME   STATUS_TYPE
 0             INACTIVE      RECORD EXPIRED
 1             ACTIVE        FULL TIME
 2             ACTIVE        PART TIME
3 rows selected
SQL> set feedback 4
SQL> select * from work_status;
 STATUS_CODE   STATUS_NAME   STATUS_TYPE
 0             INACTIVE      RECORD EXPIRED
 1             ACTIVE        FULL TIME
 2             ACTIVE        PART TIME
```

- SET LINESIZE n
  The SET LINESIZE command is synonymous with the SET LINE LENGTH command.
- SET PAGESIZE n
  The SET PAGESIZE command is synonymous with the SET PAGE LENGTH command.
- SET TIMING { ON | OFF }
  The SET TIMING enables a single line report of used CPU and Elapsed time for each successful SQL statement or command. This is shown in the following example.

```
SQL> start transaction;
SQL> set timing on;
SQL> select count(*)
cont> from employees
cont>      inner join job_history using (employee_id)
cont>      inner join salary_history using (employee_id)
cont>      inner join departments using (department_code)
cont>      inner join jobs using (job_code)
cont>      left outer join resumes using (employee_id)
cont>      left outer join degrees using (employee_id)
cont>      left outer join colleges using (college_code)
cont>
cont> ;

        3871
1 row selected
Timing: Elapsed:    0 00:00:00.82 Cpu:    0 00:00:00.16
SQL> set timing off;
SQL> commit;
```

Chapter 6
Enhancements And Changes Provided in Oracle Rdb Release 7.2.3.0

6.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.3.0

6.1.1 Optional Run−Time Routine Native Compiler on I64 Enabled By Default

On Alpha and VAX systems, Oracle Rdb generates, at run time, callable subroutines of native machine instructions. These subroutines are executed during the processing of database insert/update/retrieval operations.

Oracle Rdb on I64 systems uses a hardware−portable instruction interpreter. This allowed rapid development and deployment of Oracle Rdb on Integrity OpenVMS systems. Release 7.2.1.1 of Oracle Rdb introduced a second pass optimization that converts these portable instructions to native I64 machine code for enhanced performance. This feature is enabled by default starting with Release 7.2.3 of Oracle Rdb.

To disable the run−time compiler, define RDMS$BIND_CODE_OPTIMIZATION to a value of "0" or use the flag "CODE_OPTIMIZATION(0)". The run−time compiler is enabled by default. To re−enable, if needed, the run−time compiler on I64 systems, either deassign the logical name RDMS$BIND_CODE_OPTIMIZATION or define the logical name RDMS$BIND_CODE_OPTIMIZATION to a value of "2" or use the flag "CODE_OPTIMIZATION(2)". This logical name and flag have no effect on Alpha systems. The following examples show use of the logical name and the flag to enable and disable and show the status of the optional run−time compiler on I64 systems.

```
! To disable:
$ DEFINE RDMS$BIND_CODE_OPTIMIZATION 0
$ DEFINE RDMS$SET_FLAGS "CODE_OPTIMIZATION(0)"
SQL> SET FLAGS 'CODE_OPTIMIZATION(0)';

! To enable:
$ DEFINE RDMS$BIND_CODE_OPTIMIZATION 2
$ DEFINE RDMS$SET_FLAGS "CODE_OPTIMIZATION(2)"
SQL> SET FLAGS 'CODE_OPTIMIZATION(2)';

! Show current setting enabled and disabled:
SQL> ATT 'FI MF_PERSONNEL';
SQL> SET FLAGS 'CODE_OPTIMIZATION(2)';
SQL> SHOW FLAGS

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),REFINE_ESTIMATES(127)
   ,CODE_OPTIMIZATION(2),NOBITMAPPED_SCAN

SQL> SET FLAGS 'CODE_OPTIMIZATION(0)';
SQL> SHOW FLAGS

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

There is CPU overhead required to compile a subroutine and not every subroutine can be compiled, at present, into native Itanium routines. Additional virtual memory is required for the functionality of the optional run−time compiler.

If you detect a difference in the functionality or behavior of Oracle Rdb when this feature is enabled, you may define the logical name to revert to the prior (interpreted) execution model.

6.1.2 Temporary Table Improvements

In prior versions of Oracle Rdb, all temporary table data structures were always allocated in 32−bit process (P0) virtual address space. In several cases, the 1GB size of P0 space drastically limited the number of rows that could be stored or would result in unexpected errors when deleting or modifying rows. Additionally, performance when accessing rows within temporary tables would degrade as additional rows were inserted.

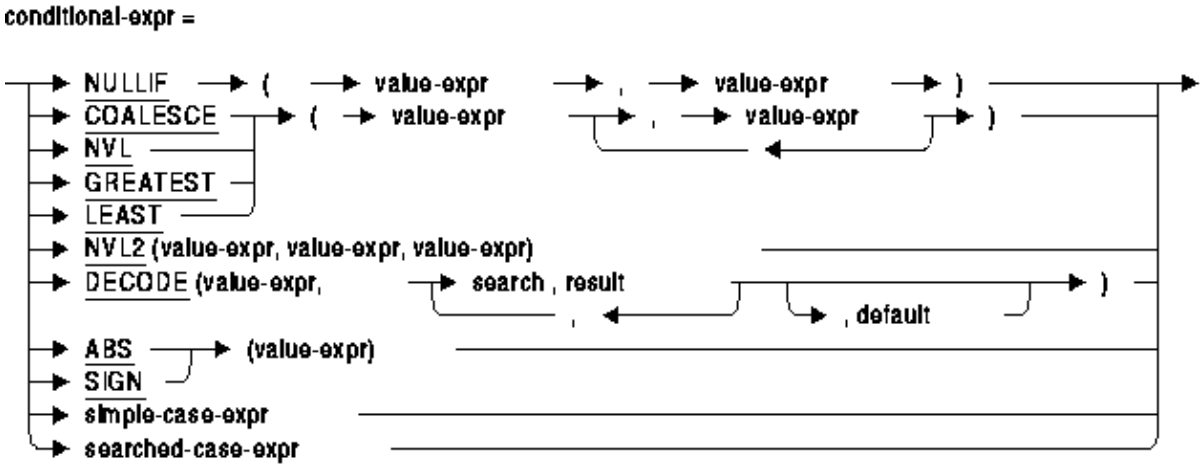
The following enhancements have been made to address these cases:

- With the exception of the actual data rows themselves, most of the data structures related to temporary tables have been moved to 64−bit process (P2) virtual address space. The data content remains in 32−bit address space.
- Use of the logical name RDMS$TTB_HASH_SIZE to control the size of an internal hash table may no longer be required for consistent performance. The hash table is now initially sized larger than it was in the past and will automatically extend as needed to accommodate the rows in the temporary table with more consistent performance.

6.1.3 New SIGN Builtin Function

This release of Oracle Rdb enhances the Conditional expressions with support for the SIGN builtin function.

Syntax



Usage Notes

- SIGN returns an INTEGER value.
- SIGN accepts any numeric (fixed or floating) or interval value expression.
- If the value expression evaluates to NULL, then the SIGN function returns NULL.
- If the value expression evaluates to a negative value, then SIGN returns −1; if the value is positive then SIGN returns 1; otherwise a zero will be returned.
- The SQL_FUNCTIONS script continues to add the SIGN function to the database. However, this function is now deprecated and is retained only for backward compatibility with applications built using that function.

Examples

This example computes delayed departures from the LAYOVER table.

Example 6−1 Using SIGN Builtin Function

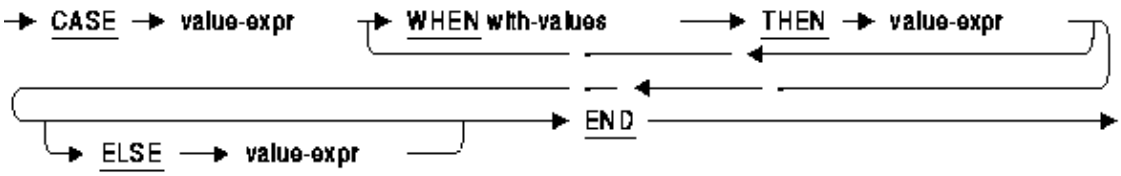
```
SQL> select arr_date,
cont>       dep_date,
cont>       DECODE (SIGN ((dep_date − arr_date) day(9)),
cont>           −1, 'date error − can not depart before arrival',
cont>            0, 'same day departure',
cont>            1, 'delayed')
cont> from LAYOVER;

 2005−12−22   2006−01−20   delayed
 2005−12−23   2005−12−25   delayed
 2006−01−30   2006−02−01   delayed
 2006−02−06   2006−02−09   delayed
 2006−01−24   2006−01−26   delayed
 2006−02−02   2006−02−19   delayed
 2007−02−10   2007−02−16   delayed
 2007−02−20   2007−02−26   delayed
 2007−05−29   2007−06−08   delayed
 2007−06−12   2007−06−26   delayed
 2007−05−15   2007−05−21   delayed
 2007−09−10   2007−09−14   delayed
 2007−09−04   2007−09−06   delayed
 2007−09−19   2007−09−20   delayed
 2007−09−21   2007−09−24   delayed
15 rows selected
```

6.1.4 Enhanced Simple CASE Expression

This release of Oracle Rdb enhances the Simple CASE expression to support value lists and partial predicates.

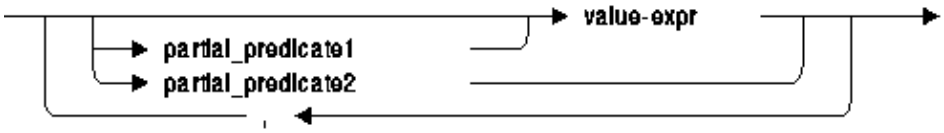
Syntax

simple-case-expr =



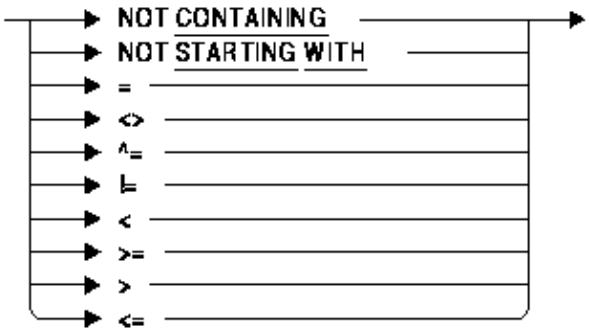
---

with-values =



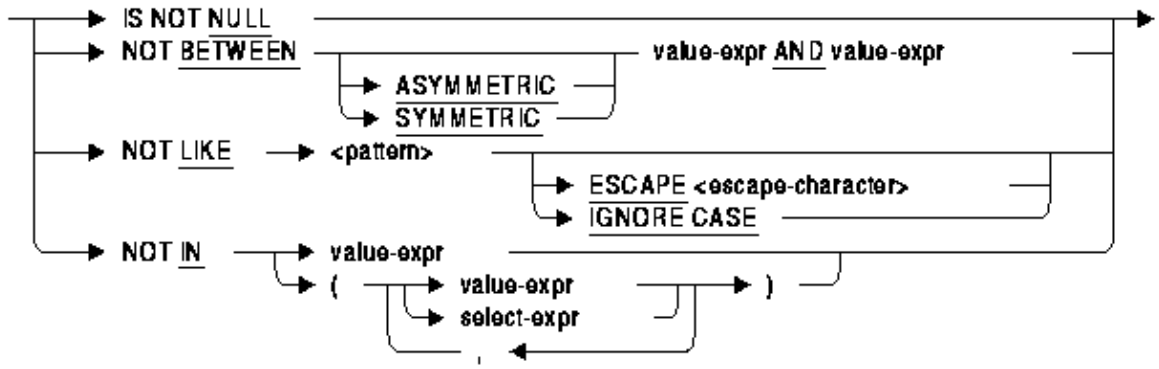
---

partial_predicate1 =



---

partial_predicate2 =



Usage Notes

- The WHEN clause can now include a list of value expressions. In prior versions, the WHEN clause was limited to just one value expression. Each value expression is separated by a comma.
- The WHEN clause can now include partial predicates. In prior versions, the default operator that was applied to the case primary expression of each WHEN value expression was equality (=). With this release, this can be overridden. Multiple partial predicates can also be listed.
  A WHEN clause partial predicate can begin with one of the following operators: IS NULL, IS NOT NULL, BETWEEN, NOT BETWEEN, IN, NOT IN, LIKE, NOT LIKE, STARTING WITH, NOT STARTING WITH, CONTAINING, NOT CONTAINING, <=, <, <>, >=, >, ^=, !=, and =.
- If the WHEN clause includes the value expression NULL, it assumes that this is equivalent to the IS NULL operator.

Examples

This example calculates the calendar quarter using a CASE expression.

Example 6−2 Using the IN clause as a partial predicate

```
SQL> select
cont>     salary_end,
cont>     case extract (month from salary_end)
cont>         when in (1, 2, 3) then 'Q1'
cont>         when in (4, 5, 6) then 'Q2'
cont>         when in (7, 8, 9) then 'Q3'
cont>         when in (10, 11, 12) then 'Q4'
cont>         else 'UNKNOWN'
cont>     end quarter
cont> from salary_history
cont> where employee_id = '00164'
cont> order by salary_start;
 SALARY_END    QUARTER
  2-Mar-1981   Q1
 21-Sep-1981   Q3
 14-Jan-1983   Q1
 NULL          UNKNOWN
4 rows selected
```

```
SQL>
```

A simple list of values could also be used to achieve the same result.

```
SQL> select
cont>     salary_end,
cont>     case extract (month from salary_end)
cont>         when 1, 2, 3 then 'Q1'
cont>         when 4, 5, 6 then 'Q2'
cont>         when 7, 8, 9 then 'Q3'
cont>         when 10, 11, 12 then 'Q4'
cont>         else 'UNKNOWN'
cont>     end quarter
cont> from salary_history
cont> where employee_id = '00164'
cont> order by salary_start;
 SALARY_END    QUARTER
  2-Mar-1981   Q1
 21-Sep-1981   Q3
 14-Jan-1983   Q1
 NULL          UNKNOWN
4 rows selected
SQL>
```

This example shows the use of partial predicates to process the LAST_NAME column and provide a customized ordering of the EMPLOYEES table.

Example 6-3 Using Partial predicates in the CASE expression

```
SQL> select
cont>     concat (trim (first_name), ' ',
cont>             nvl2 (middle_initial,
cont>                   concat (middle_initial,'.',last_name),
cont>                   last_name)) as names
cont> from employees
cont> order by
cont>     case last_name
cont>         when starting with 'Mc'
cont>         then
cont>             substring (last_name from 3)
cont>         when starting with 'Mac'
cont>         then
cont>             substring (last_name from 4)
cont>         when containing ''''
cont>         then
cont>             substring (last_name from position (''''' in last_name)+1)
cont>         else
cont>             last_name
cont>     end;
 NAMES
 Louie A.Ames
 Aruwa D'Amico
 Leslie Q.Andriola
 Joseph Y.Babbin
 Dean G.Bartlett
   .
   .
   .
```

```
 Brian Wood
 Al F.Ziemke
100 rows selected
```

This example shows the use of value lists to make the simple case more readable. Using input values for the month and year, it returns the number of days in that month.

Example 6−4 Using lists of values in the Simple CASE expression

```
SQL> set flags 'trace';
SQL> begin
cont> declare :year_val, :month_val int;
cont> set :year_val = extract (year from current_date);
cont> for :month_val in 1 to 12
cont> do
cont>     trace
cont>         case :month_val
cont>             when 1, 3, 5, 7, 8, 10, 12 then
cont>                 31
cont>             when 4, 6, 9, 11 then
cont>                 30
cont>             when 2 then
cont>                 DECODE (MOD (:year_val, 4), 0, 29, 28)
cont>             else
cont>                 NULL
cont>         end;
cont> end for;
cont> end;
~Xt: 31
~Xt: 29
~Xt: 31
~Xt: 30
~Xt: 31
~Xt: 30
~Xt: 31
~Xt: 31
~Xt: 30
~Xt: 31
~Xt: 30
~Xt: 31
```

6.1.5 Changes in Generated Query Outline ID

This release of Oracle SQL has added various optimizations to the Simple CASE expression as well as the DECODE and ABS functions. These changes might, in some cases, change the generated intermediate query. This will have no effect on the query results but will change the query id generated for a query outline. In such cases, the query outline should be re−created to ensure matching by the new outline id.

Alternately, queries using these query outlines can be modified to use the OPTIMIZE USING clause to use the name of the query outline. Then such changes as described here do not affect the selection of the query outline. If a query outline is used in conjunction with the RMU Unload command then the /OPTIMIZE=USING clause can be used to specify the name of the query outline for use by RMU.

Note

This query outline id is also displayed by the SET FLAGS 'WATCH_OPEN' flag as shown in this example:

```
SQL> set flags 'watch_open';
SQL> select decode (employee_id, '00164', 'Toliver', 'Others') from employees;
~Xo: Start Request ECB31E522CC71247B16B18660AD42F1D (unnamed)
   .
   .
   .
```

This is the query id generated by Oracle Rdb V7.1.

```
SQL> select decode (employee_id, '00164', 'Toliver', 'Others') from employees;
~Xo: Start Request 77D353270F7842D60FCA3D6CC11378D3 (unnamed)
   .
   .
   .
```

6.1.6 ALTER INDEX ... MAINTENANCE IS ENABLED DEFERRED Syntax is Now Active

Bug 6880752

In prior versions of Oracle Rdb, the ALTER INDEX ... MAINTENANCE IS ENABLED DEFERRED syntax was not active. Instead it produced a WISH_LIST error. This clause has now been activated for this release.

The following example shows the WISH_LIST error generated in prior versions.

```
SQL> alter index t1_idx maintenance is enabled deferred;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-F-WISH_LIST, feature has not been implemented
```

When ALTER INDEX ... MAINTENANCE IS ENABLED DEFERRED is used on an index, it has the following actions:

- If the index is MAINTENANCE IS ENABLED IMMEDIATE mode, that index is changed to a build−pending index. The side effect of this change is that the table upon which the index is defined can now only be read. The index partitions may now be rebuilt using REBUILD PARTITION or a combination of TRUNCATE PARTITION and BUILD PARTITION clauses.
  A warning message is issued upon successful execution of the ALTER INDEX command so that the database administrator is aware that the index is incomplete and also that the table upon which the index is defined is write−disabled so that INSERT, UPDATE and DELETE statements cannot be used.

  ```
  SQL> alter index t1_idx maintenance is enabled deferred;
  %RDB-W-META_WARN, metadata successfully updated with the reported warning
  -RDMS-W-IDXBLDPEND, index in build pending state - maintenance is disabled
  SQL>
  ```

After each partition is rebuilt, a final ALTER INDEX ... MAINTENANCE IS ENABLED IMMEDIATE statement will be required to make the index active and allow updates to the table. This final step validates all partitions, rolls up partition cardinality information and clears the build−pending state.

- If the index is MAINTENANCE IS ENABLED DEFERRED mode then there are no changes made to the index and the ALTER INDEX statement quietly succeeds.
- If the index is MAINTENANCE IS DISABLED then an error is raised. This ALTER INDEX operation is not compatible with such indices. However, ALTER INDEX ... TRUNCATE ALL PARTITIONS or ALTER INDEX ... TRUNCATE PARTITION can be used to modify a disabled index into a build−pending index.

```
SQL> alter index t1_idx maintenance is enabled deferred;
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−E−INDMAINTDIS, maintenance on index T1_IDX has been disabled
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.3.

6.1.7 SQL Precompiler and Module Language Compiler /ARCHITECTURE Command Line Qualifier

For improved performance of generated code, the SQL Language Precompiler and Module Language Compilers include support for the /ARCHITECTURE=keyword command line qualifier on OpenVMS Alpha systems. At present, the /ARCHITECTURE qualifier is ignored on Itanium systems.

The /ARCHITECTURE=keyword qualifier specifies the lowest version of the Alpha architecture where this code will run which can allow the compiler to generate more efficient code with the tradeoff that code may not run on older systems.

All Alpha processors implement a core set of instructions and, in some cases, the following extensions:

- Byte/word extension (BWX) − The instructions that comprise the BWX extension are LDBU, LDWU, SEXTB, SEXTW, STB, and STW.
- Square−root and floating−point convert extension (FIX) − The instructions that comprise the FIX extension are FTOIS, FTOIT, ITOFF, ITOFS, ITOFT, SQRTF, SQRTG, SQRTS, and SQRTT.
- Count extension (CIX) − The instructions that comprise the CIX extension are CTLZ, CTPOP, and CTTZ.
- Multimedia extension (MVI) − The instructions that comprise the MVI extension are MAXSB8, MAXSW4, MAXUB8, MAXUW4, MINSB8, MINSW4, MINUB8, MINUW4, PERR, PKLB, PKWB, UNPKBL, and UNPKBW.

The Alpha Architecture Reference Manual describes the extensions in detail.

The keyword specified with the /ARCHITECTURE qualifier determines which instructions the compiler can generate and which coding rules it must follow.

- GENERIC − Generate instructions that are appropriate for all Alpha processors. This option is the default and is equivalent to /ARCH=EV4.
- HOST − Generate instructions for the processor that the compiler is running on (for example, EV56

instructions on an EV56 processor, EV7 instructions on an EV7 processor, and so on).

- EV4 − Generate instructions for the EV4 processor (21064, 20164A, 21066, and 21068 chips). Applications compiled with this option will not incur any emulation overhead on any Alpha processor.
- EV5 − Generate instructions for the EV5 processor (some 21164 chips). (Note that the EV5 and EV56 processors both have the same chip number − 21164.) Applications compiled with this option will not incur any emulation overhead on any Alpha processor.
- EV56 − Generate instructions for EV56 processors (some 21164 chips). This option permits the compiler to generate any EV4 instruction, plus any instructions contained in the BWX extension. Applications compiled with this option may incur emulation overhead on EV4 and EV5 processors.
- PCA56 − Generate instructions for PCA56 processors (21164PC chips). This option permits the compiler to generate any EV4 instruction plus any instructions contained in the BWX and MVI extensions. Applications compiled with this option may incur emulation overhead on EV4 and EV5 processors.
- EV6 − Generate instructions for EV6 processors (21264 chips). This option permits the compiler to generate any EV4 instruction, any instruction contained in the BWX and MVI extensions, plus any instructions added for the EV6 chip. These new instructions include a floating−point square root instruction (SQRT), integer/floating−point register transfer instructions, and additional instructions to identify extensions and processor groups. Applications compiled with this option may incur emulation overhead on EV4, EV5, EV56, and PCA56 processors.
- EV67 or EV68 − Generate instructions for EV67 and EV68 processors (21264A chips). This option permits the compiler to generate any EV6 instruction, plus the new bit count instructions (CTLZ, CTPOP, and CTTZ). However, the precompilers do not currently generate any of the new bit count instructions and the EV67 and EV68 have identical instruction scheduling models so the EV67 and EV68 are essentially identical to the EV6. Applications compiled with this option may incur emulation overhead on EV4, EV5, EV56, and PCA56 processors.
- EV7 − Generate instructions for the EV7 processor (21364 chip). This option permits the compiler to generate any EV67 instruction. There are no additional instructions available on the EV7 processor but the compiler does have different instruction scheduling and prefetch rules for tuning code for the EV7. Applications compiled with this option may incur emulation overhead on EV4, EV5, EV56, and PCA56 processors.

The OpenVMS Alpha operating system includes an instruction emulator. This capability allows any Alpha chip to execute and produce correct results from Alpha instructions − even if some of the instructions are not implemented on the chip. Applications using emulated instructions will run correctly but may incur significant emulation overhead at run time.

Of the available extension types, the Byte/word extension (BWX) will often be beneficial for increased performance of Rdb−based applications. In addition, for those Alpha implementations that support quad−issue of instructions (the EV6 and later processors), the compiler does have different instruction scheduling and prefetch rules for tuning code.

For highest levels of performance of generated code, Oracle recommends that the /ARCHITECTURE switch be specified with the keyword that most closely matches the lowest processor type of the machine where the program will execute.

Language Compiler Support for /ARCHITECTURE

If specified, the /ARCHITECTURE qualifier is passed on the command line to the specified language compiler by the SQL Precompiler. The language compiler being used must support the /ARCHITECTURE qualifier and the architecture keyword value when the

/ARCHITECTURE qualifier is specified.

6.1.8 PERFT4_RDB Example Program

Accessing performance information in a tabular fashion for Oracle Rdb databases can be often be beneficial. In particlar, data in "CSV" (comma separated values) format can be readily loaded into other applications. HP's T4 utility program TlViz expects data in an enhanced CSV format where the first 4 lines of the data file contain additional information.

SQL$SAMPLE:PERFT4_RDBxx.C is a sample C program that reads an RMU /SHOW STATISTICS binary file and converts all statistic values for each sample into a current rate per second. The statistics values are written to an output text file. This example is supplied as a template for writing your own program.

To use the PERFT4_RDB program, optionally compile and link (the example program has been supplied also as a .EXE) and then create a foreign command symbol with a value of "SQLSAMPLE:PERFT4_RDBxx.EXE" (where xx is the version of Rdb) and pass an input binary statistics file name and an output text file name. The following example command sequence demonstrates one possible way that statistics can be gathered for one hour and then formatted.

```
$ PERFT4 := $SQL$SAMPLE:PERFT4_RDB72
$ RMU /SHOW STATISTICS MFP −
        /NOINTERACTIVE −
        /OUTPUT = 2008−11−16−00−56.STATS −
        /UNTIL = "16−NOV−2008 11:00:00" −
        /TIME = 15
$ PERFT4 2008−11−16−00−56.STATS 2008−11−16−00−56.CSV
Wrote 251 records (from 16−Nov−2008 09:56:29 to 16−Nov−2008 10:59:57)
```

This example source code SQL$SAMPLE:PERFT4_RDBxx.C is intended solely to be used as a template for writing your own program. No support for this example template program is expressed or implied. Oracle Corporation assumes no responsibility for the functionality, correctness or use of this example program. Oracle Corporation reserves the right to change the format and contents of the Oracle Rdb RMU SHOW STATISTICS binary output file at any time without prior notice.

6.1.9 RMU Load Quietly Truncated String Data During Insert

Bug 6732438

In prior versions of Oracle Rdb, the RMU Load command would quietly truncate data loaded into CHAR and VARCHAR columns. This loss of data might be significant but was never reported by Oracle Rdb.

This problem has been corrected in Oracle Rdb Release 7.2.3. RMU Load now defaults to SQL dialect SQL99 which implicitly checks for and reports truncations during INSERT operations.

This behavior is controlled by a new /DIALECT qualifier.

- /NODIALECT, /DIALECT=SQL89 or /DIALECT=NONE will revert to prior behavior and no truncation error will be reported.
- /DIALECT=SQL99 (the default) will enable reporting of truncation errors. Note that truncation occurs if non−space characters are discarded during the insert.

```
$ rmu/load abc f2 f1
%RMU-I-LOADERR, Error loading row 1.
%RDB-E-TRUN_STORE, string truncated during assignment to a column
%RMU-I-DATRECREAD,  1 data records read from input file.
%RMU-I-DATRECSTO,   0 data records stored.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 13-FEB-2008 15:39:44.40
$
```

6.1.10 New FETCH FIRST and OFFSET Clauses for Select Expression

This release of Oracle Rdb adds two new clauses to the SELECT expression. These clauses are defined by the draft SQL Database Language Standard 2008.

- OFFSET skip-expression
  The OFFSET clause allows the database programmer to start fetching the result rows from the specified offset within the result table. OFFSET accepts a numeric value expression which may contain arbitrary arithmetic operators, function calls, subselect clauses or sequence references. The subselect clauses may not reference columns in the outer query as it is evaluated before row processing begins.

  Note

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

  The OFFSET clause is equivalent in functionality to the SKIP clause currently supported by the LIMIT TO clause. The distinction is that OFFSET can be specified without a row limit.
  OFFSET is not compatible with the SKIP (or OFFSET) sub clause of the LIMIT TO clause. However, OFFSET and LIMIT TO can be used together.
- FETCH FIRST limit-expression
- FETCH NEXT limit-expression
  The FETCH FIRST clause allows the database programmer to limit the results returned from a query expression. The FETCH FIRST clause is equivalent to functionality currently supported by the LIMIT TO clause. FETCH accepts a numeric value expression which may contain arbitrary arithmetic operators, function calls, subselect clauses or sequence references. The subselect clauses may not reference columns in the outer query as it is evaluated before row processing begins.
  The FETCH NEXT is identical to FETCH FIRST but allows the syntax to be more descriptive when coupled with the OFFSET clause.
  If no value expression is provided for FETCH, it will default to 1 row.
  The FETCH clause is not compatible with the LIMIT TO clause.

Examples

The following examples show the use of the FETCH FIRST and OFFSET clauses.

This example uses the DEPARTMENTS table to locate the employee id of each manager and, after sorting them by their birthday, the oldest manager's name and employee id are displayed.

Example 6−5 Using FETCH FIRST to find the oldest manager in the company

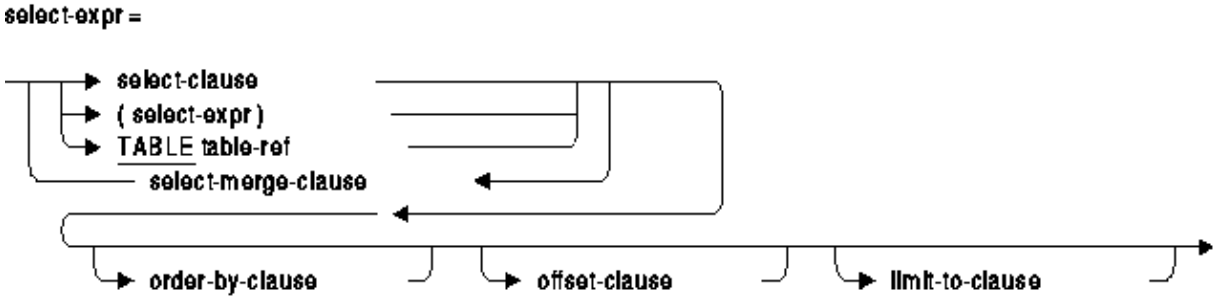
```
SQL> -- select the most senior manager
SQL> select e.last_name, e.first_name, e.employee_id
cont> from departments d, employees e
cont> where d.manager_id = e.employee_id
cont> order by e.birthday
cont> fetch first row only;
 E.LAST_NAME      E.FIRST_NAME   E.EMPLOYEE_ID
 O'Sullivan       Rick           00190
1 row selected
SQL>
```

This query uses a subselect in the OFFSET clause to locate the median (or middle) row of the sorted set.

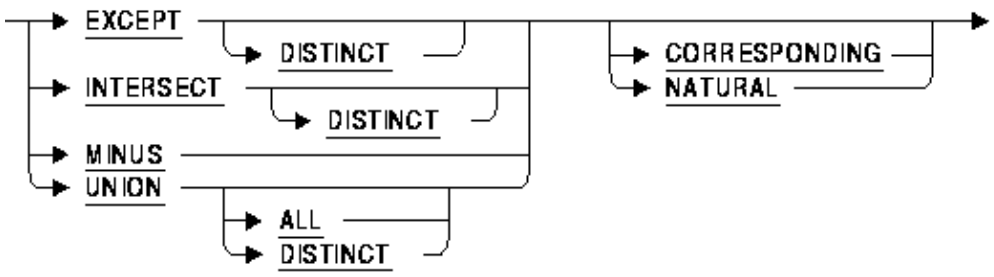
Example 6−6 Using OFFSET ROWS and FETCH NEXT to compute the median salaried employee

```
SQL> select e.last_name, e.first_name, employee_id, sh.salary_amount
cont> from salary_history sh inner join employees e using (employee_id)
cont> where sh.salary_end is null
cont> order by sh.salary_amount
cont> offset (select count(*)
cont>         from salary_history
cont>         where salary_end is null)/2 rows
cont> fetch next row only;
 E.LAST_NAME      E.FIRST_NAME   EMPLOYEE_ID   SH.SALARY_AMOUNT
 Boyd             Ann            00244               $24,166.00
1 row selected
SQL>
```

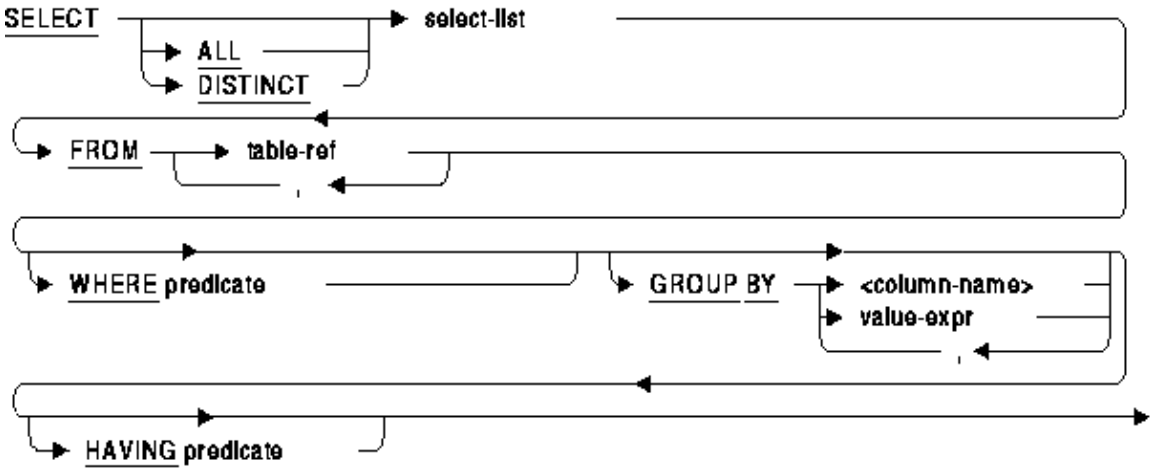
Syntax



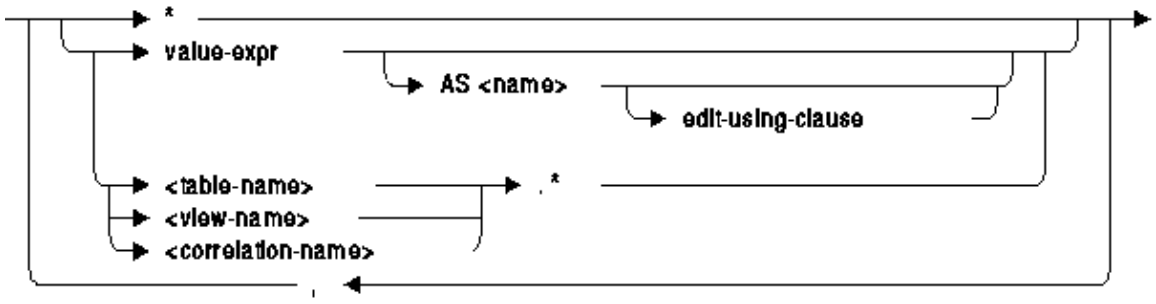
select-merge-clause =



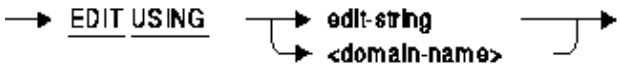
select-clause =



select-list =

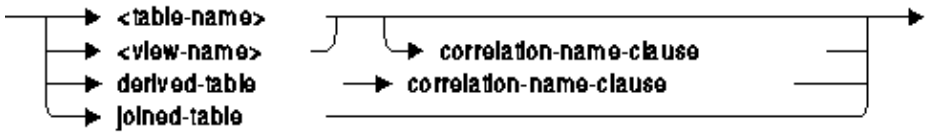


edit-using-clause =



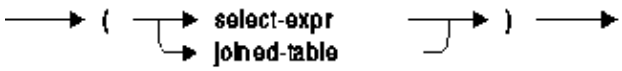
---

table-ref =



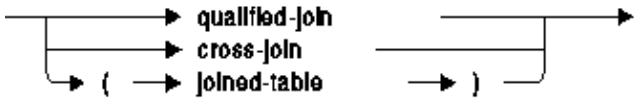
---

derived-table =



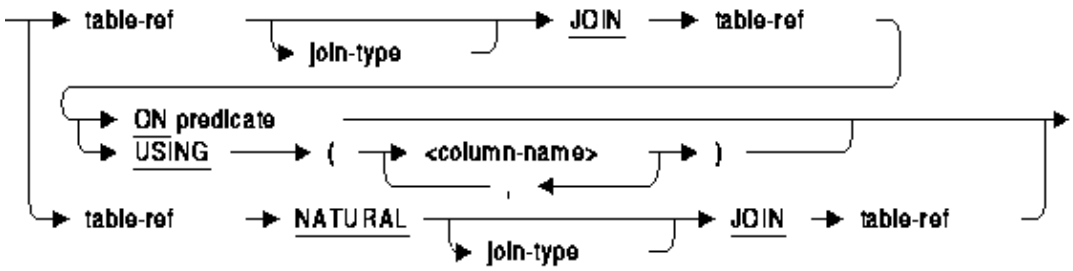
---

joined-table =



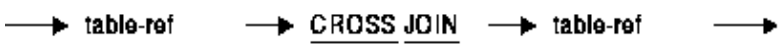
---

qualified-join =



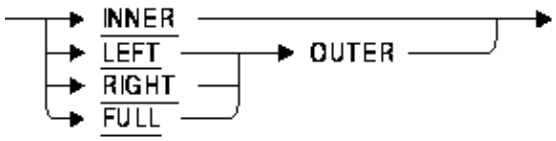
---

cross-join =



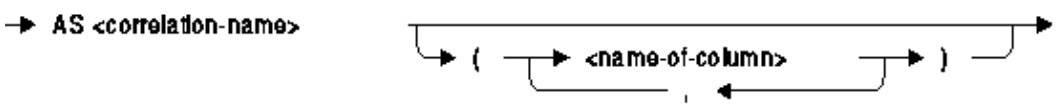
---

join-type =



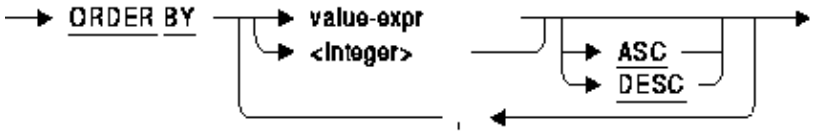
---

correlation-name-clause =



---

order-by-clause =

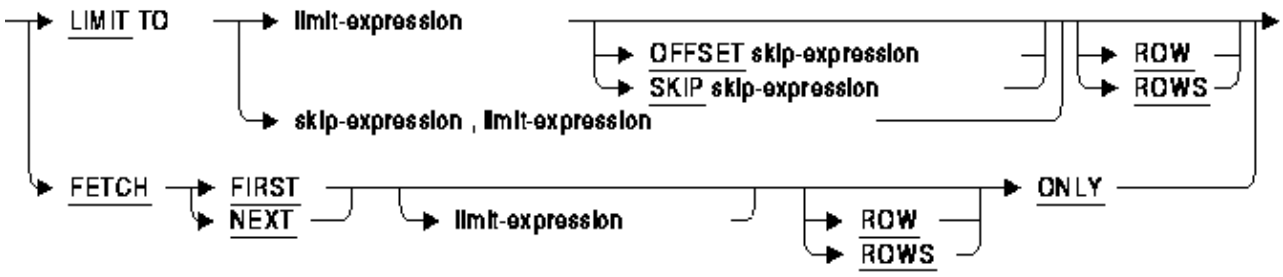


---

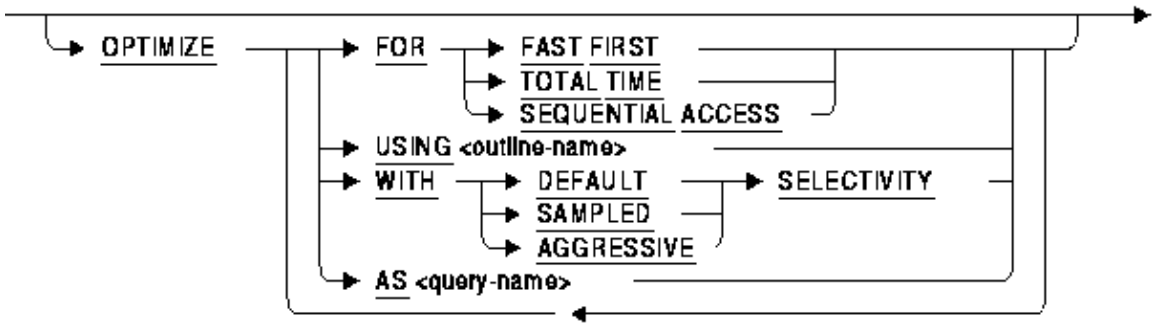
offset-clause =

---

limit-to-clause =



optimize-clause =



Usage Notes

- If ORDER BY is used in a query that includes OFFSET, FETCH FIRST (FETCH NEXT), or LIMIT TO clauses then the rows are first retrieved and sorted prior to applying the OFFSET, FETCH or LIMIT TO actions.
- A select expression may contain both OFFSET and FETCH NEXT (LIMIT TO) clauses, in which case the OFFSET is applied first and then the FETCH NEXT clause. For instance, if the query would normally return 100 rows, then an OFFSET 20 would skip over the first 20 rows and return the remaining 80. If on the other hand an OFFSET 20 and a LIMIT TO 20 were specified, then after skipping the first 20 rows the next 20 are returned.
- The OFFSET, FETCH FIRST or LIMIT TO clauses may result in no rows being retrieved.

6.1.11 RMU Unload Now Creates SQL*Loader Control Files

Enhancement 2146782

This release of Oracle Rdb has enhanced RMU Unload by providing support for SQL*Loader control files and portable data files.

The following example shows the FORMAT=CONTROL option.

```
$ RMU/UNLOAD/RECORD_DEFINITION=(FORMAT=CONTROL,FILE=EMP) -
    SQL$DATABASE -
    EMPLOYEES -
    EMPLOYEES
```

This command creates a file EMP.CTL (the SQL*Loader control file) and EMPLOYEES.DAT in a portable format to be loaded.

Usage Notes

- FORMAT=CONTROL implicitly uses a portable data format as TEXT rather than binary values. The unloaded data files are similar to that generated by FORMAT=TEXT but include a NULL vector to represent NULL values ('1') and non−NULL values ('0').
  The SQL*Loader control file uses this NULL vector to set NULL for the data upon loading.
- When FORMAT=CONTROL is used, the output control file and associated data file are intended to be used with the Oracle RDBMS SQL*Loader (sqlldr) command to load the data into an Oracle RDBMS database table. LIST OF BYTE VARYING (SEGMENTED STRING) columns are not unloaded.
  The file specification for the FILE option will default to a .CTL file type.
  Oracle Rdb does not support this format for RMU Load.
- The keywords NULL, PREFIX, SEPARATOR, SUFFIX, and TERMINATOR only apply to DELIMITED_TEXT format and may not be used in conjunction with the CONTROL keyword.
- DATE VMS data is unloaded including the fractional seconds precision. However, when mapped to Oracle DATE type in the control file, the fractional seconds value is ignored. It is possible to modify the generated control file to use the TIMESTAMP type and add FF to the date edit mask.

Chapter 7
Documentation Corrections, Additions and Changes

This chapter provides corrections for documentation errors and omissions.

7.1 Documentation Corrections

7.1.1 Back Out SQL_PROTOCOL_RETRY Configuration Parameter

In older versions of Oracle Rdb, remote connections would retry the connection using the old V5.1 protocol following a failed remote connection attempt. This feature was by default disabled in Oracle Rdb V7.2.2 but has now been enabled again. The configuration file parameter keyword SQL_PROTOCOL_RETRY has been removed. This is because no positive effect was seen from this change.

This change was originally documented in the Oracle Rdb 7.2.2 New Features Chapter but has since been removed. More work will be done in this area in the future.

7.1.2 Revised Example for SET OPTIMIZATION LEVEL Statement

Bug 6350960

Example 1: Setting the optimization level

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

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

```
SQL> set flags 'STRATEGY,DETAIL';
SQL> --
SQL> -- No optimization level has been selected. The optimizer
SQL> -- selects the FAST FIRST (FFirst) retrieval tactic to
SQL> -- retrieve the rows from the EMPLOYEES table in the
SQL> -- following query:
SQL> --
SQL> select EMPLOYEE_ID, LAST_NAME
cont> from EMPLOYEES
cont> where EMPLOYEE_ID IN ('00167', '00168');
Tables:
  0 = EMPLOYEES
Leaf#01 FFirst 0:EMPLOYEES Card=100
  Bool: (0.EMPLOYEE_ID = '00167') OR (0.EMPLOYEE_ID = '00168')
  BgrNdx1 EMPLOYEES_HASH [(1:1)2] Fan=1
    Keys: r0: 0.EMPLOYEE_ID = '00168'
          r1: 0.EMPLOYEE_ID = '00167'
 EMPLOYEE_ID   LAST_NAME
 00167         Kilpatrick
```

```
 00168         Nash
2 rows selected
SQL> --
SQL> -- Use the SET OPTIMIZATION LEVEL statement to specify that
SQL> -- you want the TOTAL TIME (BgrOnly) retrieval strategy to
SQL> -- be used.
SQL> --
SQL> SET OPTIMIZATION LEVEL 'TOTAL TIME';
SQL> select EMPLOYEE_ID, LAST_NAME
cont> from EMPLOYEES
cont> where EMPLOYEE_ID IN ('00167', '00168');
Tables:
  0 = EMPLOYEES
Leaf#01 BgrOnly 0:EMPLOYEES Card=100
  Bool: (0.EMPLOYEE_ID = '00167') OR (0.EMPLOYEE_ID = '00168')
  BgrNdx1 EMPLOYEES_HASH [(1:1)2] Fan=1
    Keys: r0: 0.EMPLOYEE_ID = '00168'
          r1: 0.EMPLOYEE_ID = '00167'
 EMPLOYEE_ID   LAST_NAME
 00167         Kilpatrick
 00168         Nash
2 rows selected
SQL> --
SQL> -- When the SET OPTIMIZATION LEVEL 'DEFAULT' statement
SQL> -- is specified the session will revert to the default FAST FIRST
SQL> -- optimizer tactic.
SQL> --
SQL> SET OPTIMIZATION LEVEL 'DEFAULT';
SQL> select EMPLOYEE_ID, LAST_NAME
cont> from EMPLOYEES
cont> where EMPLOYEE_ID IN ('00167', '00168');
Tables:
  0 = EMPLOYEES
Leaf#01 FFirst 0:EMPLOYEES Card=100
  Bool: (0.EMPLOYEE_ID = '00167') OR (0.EMPLOYEE_ID = '00168')
  BgrNdx1 EMPLOYEES_HASH [(1:1)2] Fan=1
    Keys: r0: 0.EMPLOYEE_ID = '00168'
          r1: 0.EMPLOYEE_ID = '00167'
 EMPLOYEE_ID   LAST_NAME
 00167         Kilpatrick
 00168         Nash
2 rows selected
SQL>
```

7.1.3 RMU /VERIFY Process Quotas and Limits Clarification

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

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

7.1.4 Online Backup Can Be Performed With Transfer Via Memory

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

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

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

7.1.5 Missing Example for CREATE STORAGE MAP

Bug 5655348

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

Example

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

```
SQL> create database
cont>     filename 'DB$:MULTIMEDIA'
cont>
cont>     create storage area PHOTO_AREA1
cont>         filename 'DB$:PHOTO_AREA1'
cont>         page format UNIFORM
cont>
cont>     create storage area PHOTO_AREA2
cont>         filename 'DB$:PHOTO_AREA2'
cont>         page format UNIFORM
cont>
cont>     create storage area TEXT_AREA
cont>         filename 'DB$:TEXT_AREA'
cont>         page format UNIFORM
cont>
cont>     create storage area AUDIO_AREA
cont>         filename 'DB$:AUDIO_AREA'
cont>         page format UNIFORM
cont>
cont>     create storage area DATA_AREA
cont>         filename 'DB$:DATA_AREA'
cont>         page format UNIFORM
cont> ;
SQL>
SQL> create table EMPLOYEES
```

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

```
            fill randomly
        in RDB$SYSTEM
                (partition SYSTEM_LARGE_OBJECTS)

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

7.1.6 RDM$BIND_MAX_DBR_COUNT Documentation Clarification

Bugs 1495227 and 3916606

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

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

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

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

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

failure" recovery.

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

Per−Database Value

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

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

7.1.7 Database Server Process Priority Clarification

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

Individual server priorities can be explicitly controlled via system−wide logical names as described in Table 7−1.

Table 7−1 Server Process Priority Logical Names

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

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

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

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

Bug 2500594

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

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

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

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

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

For example:

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

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

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

Now run SQL and verify that the version is correct:

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

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

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

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

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

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

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

7.1.9 Clarification of PREPARE Statement Behavior

Bug 2581863

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

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

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

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

7.1.10 RDM$BIND_LOCK_TIMEOUT_INTERVAL Overrides the Database Parameter

Bug 2203700

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

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

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

RDM$BIND_LOCK_TIMEOUT_INTERVAL is defined, the value of this logical name overrides the value specified in CREATE DATABASE.

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

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

7.2 Address and Phone Number Correction for Documentation

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

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

7.3 Online Document Format and Ordering Information

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

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

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

Customers should contact their Oracle representative to purchase printed documentation.

Chapter 8
Known Problems and Restrictions

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

8.1 Known Problems and Restrictions in All Interfaces

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

8.1.1 Standalone WITH Clause in Compound Statements Now Deprecated

In prior versions of Oracle Rdb, it was permitted to follow the BEGIN keyword in a top level compound statement or stored routine with a WITH HOLD clause to specify that the procedure treated all FOR loops as HOLD cursors.

Unfortunately, this syntax conflicts with recent syntax added to the ANSI and ISO SQL Database Language standard. Support for this new syntax (known as subquery factoring) is used by Oracle tools accessing Oracle Rdb through OCI Services for Rdb. Therefore, to accommodate this change Oracle will remove the WITH HOLD syntax as a standalone clause after the BEGIN keyword. The alternate syntax, available since Oracle Rdb V7.1, is to use the PRAGMA clause which allows the WITH HOLD clause to be specified.

Any applications or SQL command files must be modified prior to the next release of Oracle Rdb V7.2 after Release 7.2.3.2. At which time, the old syntax will be removed and the new WITH clause (aka subquery factoring) will be introduced.

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

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

It should be replaced with the following syntax which provides the same behavior.

```
SQL> begin
cont> pragma (with hold preserve none)
cont> trace 'a';
cont> end;
```

8.1.2 Calling DECC$CRTL_INIT

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

DECC$CRTL_INIT is a C run time library routine that allows developers to call the HP C RTL from other languages or to use the HP C RTL when the main function is not in C. It initializes the run−time environment and establishes both an exit and condition handler. The Oracle Rdb main images are not written in C and

should not be expected to have called DECC$CRTL_INIT prior to the user's code being invoked. The requirement for DECC$CRTL_INIT in certain cases exists in all versions of Oracle Rdb.

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

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

8.1.3 Application and Oracle Rdb Both Using SYS$HIBER

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

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

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

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

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

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

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

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

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

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

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

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

8.1.4 Unexpected RCS Termination

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

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

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

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

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

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

This recommendation applies to all versions of Oracle Rdb.

8.1.5 Possible Incorrect Results When Using Partitioned Descending Indexes on I64

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

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

```
SQL> CREATE DATABASE FILE FOO
cont>        CREATE STORAGE AREA FOOA
cont>        CREATE STORAGE AREA FOOB;
SQL>
SQL> CREATE TABLE MESA (ID INTEGER, M4 CHAR (1), M5 INTEGER);
SQL> CREATE TABLE RASA (ID INTEGER, R4 CHAR (1), R5 INTEGER);
SQL>
SQL> INSERT INTO MESA (ID, M4, M5) VALUES  (1, 'M', 1 );
1 row inserted
SQL> INSERT INTO RASA (ID, R4, R5) VALUES  (1, 'M', 1 );
1 row inserted
SQL>
SQL> CREATE INDEX X4 ON MESA (ID ASC , M4 DESC)
cont>        STORE USING (ID, M4)
cont>        IN FOOA WITH LIMIT OF (1, 'G')
cont>        OTHERWISE IN FOOB ;
SQL>
SQL> CREATE INDEX Y4 ON RASA (ID ASC , R4 DESC)
cont>        STORE USING (ID, R4)
cont>        IN FOOA WITH LIMIT OF (1, 'G' )
cont>        OTHERWISE IN FOOB ;
SQL>
SQL> COMMIT;

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

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

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

8.1.6 Response 'QUIT' to RMU Restart Prompt Loops

Bug 6001187

On Itanium, responding to a restart prompt with 'QUIT' causes the RMU command to loop creating bugchecks.

This problem does not appear on Alpha and shall be fixed in a future release on Itanium.

8.1.7 Changes for Processing Existence Logical Names

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

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

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

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

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

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

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

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

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

Bug 2351258

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

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

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

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

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

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

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

8.1.10 External Routine Images Linked with PTHREAD$RTL

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

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

The shareable image RDB$NATCONN_FUNC72.EXE supplied with OCI Services for Oracle Rdb (part of SQL/Services) is one such shareable image that is linked with PTHREAD$RTL. Customer built applications

that utilize External Routines from the RDB$NATCONN_FUNC72.EXE image must ensure that the main image is linked with PTHREAD$RTL. The external routines that a user may call that use functions from RDB$NATCONN_FUNC72.EXE include:

- TO_CHAR
- TO_NUMBER
- TO_DATE

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

8.1.11 SQL Procedure External Location Should Be Upper Case

Bug 4722422

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

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

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

```
create procedure sys$asctim(
    out :timlen smallint by reference,
    out :timbuf char(23) by descriptor,
    in  :timadr date vms by reference,
    in  :cvtflag integer by value);
    external location 'SYS$SHARE:SYS$PUBLIC_VECTORS.EXE'
    language general general parameter style;

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

8.1.12 Using Databases from Releases Earlier than V7.0

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

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

8.1.13 Partitioned Index with Descending Column and Collating Sequence

Bug 2797443

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

The following example can be used to demonstrate the problem.

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

set flags 'strategy';

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

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

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

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

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

Bug 3735144

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

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

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

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

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

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

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

There are three potential workarounds:

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

protocol stack then a command like the following could be used on the local node:

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

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

8.1.15 ILINK−E−INVOVRINI Error on I64

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

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

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

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

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

Bug 2676851

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

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

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

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

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

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

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

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

Galaxy Reboot Required

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

8.1.18 Oracle Rdb and OpenVMS ODS−5 Volumes

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

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

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

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

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

8.1.19 Optimization of Check Constraints

Bug 1448422

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

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

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

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

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

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

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

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

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

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

or:

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

In both cases the retrieval strategy will look like this:

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

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

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

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

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

The strategy for a delete on T2 is now:

```
SQL> delete from t2 where f2=1;
Aggregate-F1    Index only retrieval of relation T1
  Index name  I1 [1:1]
```

```
Temporary relation      Get     Retrieval by index of relation T2
  Index name  I2 [1:1]
%RDB-E-TRIG_INV_UPD, invalid update; encountered error condition defined for
trigger
-RDMS-E-TRIG_ERROR, trigger T2_DELETE forced an error
```

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

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

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

8.1.20 Carryover Locks and NOWAIT Transaction Clarification

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

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

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

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

- Nonrepeatable read operations: Allows the return of different results within a single transaction when an SQL operation reads the same row in a table twice. Nonrepeatable reads can occur when another transaction modifies and commits a change to the row between transactions. Because the standby database will update the data when it confirms a transaction has been committed, it is very possible to

see an SQL operation on a standby database return different results.

- Phantom read operations: Allows the return of different results within a single transaction when an SQL operation retrieves a range of data values (or similar data existence check) twice. Phantoms can occur if another transaction inserted a new record and committed the insertion between executions of the range retrieval. Again, because the standby database may do this, phantom reads are possible.

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

8.1.22 Row Cache Not Allowed While Hot Standby Replication is Active

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

8.1.24 Control of Sort Work Memory Allocation

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

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

The logical names are:

Table 8−1 Sort Memory Logicals

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

8.1.25 The Halloween Problem

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

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

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

CONCURRENCY/CONSISTENCY options in RDO/RDML.

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

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

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

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

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

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

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

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

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

problem.

8.2 SQL Known Problems and Restrictions

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

8.2.1 SET FLAGS CRONO_FLAG Removed

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

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

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

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

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

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

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

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

This restriction does not apply to interactive or dynamic SQL.

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

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

```
9
10 PROCEDURE b (SQLCODE);
11 BEGIN
12 LOCK TABLE employees FOR EXCLUSIVE WRITE MODE;
13 END;
```

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

8.2.4 Multistatement or Stored Procedures May Cause Hangs

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

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

```
Session 1:

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

Session 2:

$ RMU/BACKUP/LOG/ONLINE MF_PERSONNEL MF_PERSONNEL

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

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

```
Resource: nowait signal

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

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

8.2.5 Use of Oracle Rdb from Shareable Images

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

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

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

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

8.3 Oracle RMU Known Problems and Restrictions

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

8.3.1 RMU Convert Fails When Maximum Relation ID is Exceeded

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

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

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

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

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

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

8.3.2 RMU Unload /After_Journal Requires Accurate AIP Logical Area Information

The RMU Unload /After_Journal command uses the on−disk area inventory pages (AIPs) to determine the appropriate type of each logical area when reconstructing logical dbkeys for records stored in mixed−format

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

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

EMPLOYEES /TYPE=TABLE

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

EMPLOYEES /AREA=EMPID_OVER /TYPE=TABLE

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

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

  Note

  This type should NOT be used for the RDB$SYSTEM logical areas. This type does NOT identify system relations.

- BLOB
  Specifies that the logical area is a BLOB repository.

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

8.3.3 Do Not Use HYPERSORT with RMU Optimize After_Journal Command

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

performance for most sort and merge operations.

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

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

8.3.4 Changes in EXCLUDE and INCLUDE Qualifiers for RMU Backup

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

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

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

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

8.3.5 RMU Backup Operations Should Use Only One Type of Tape Drive

When using more than one tape drive for an RMU Backup command, all of the tape drives must be of the same type (for example, all the tape drives must be TA90s or TZ87s or TK50s). Using different tape drive types (for example, one TK50 and one TA90) for a single database backup operation may make database restoration difficult or impossible.

Oracle RMU attempts to prevent using different tape drive densities during a backup operation, but is not able to detect all invalid cases and expects that all tape drives for a backup are of the same type.

As long as all of the tapes used during a backup operation can be read by the same type of tape drive during a restore operation, the backup is likely valid. This may be the case, for example, when using a TA90 and a TA90E.

Oracle Corporation recommends that, on a regular basis, you test your backup and recovery procedures and environment using a test system. You should restore the database and then recover using AIJs to simulate failure recovery of the production system.

Consult the Oracle Rdb7 Guide to Database Maintenance, the Oracle Rdb7 Guide to Database Design and Definition, and the Oracle RMU Reference Manual for additional information about Oracle Rdb backup and restore operations.

8.3.6 RMU/VERIFY Reports PGSPAMENT or PGSPMCLST Errors

RMU/VERIFY may sometimes report PGSPAMENT or PGSPMCLST errors when verifying storage areas. These errors indicate that the Space Area Management (SPAM) page fullness threshold for a particular data page does not match the actual space usage on the data page. For a further discussion of SPAM pages, consult the Oracle Rdb7 Guide to Database Maintenance.

In general, these errors will not cause any adverse affect on the operation of the database. There is potential for space on the data page to not be totally utilized, or for a small amount of extra I/O to be expended when searching for space in which to store new rows. But unless there are many of these errors then the impact should be negligible.

It is possible for these inconsistencies to be introduced by errors in Oracle Rdb. When those cases are discovered, Oracle Rdb is corrected to prevent the introduction of the inconsistencies. It is also possible for these errors to be introduced during the normal operation of Oracle Rdb. The following scenario can leave the SPAM pages inconsistent:

1. A process inserts a row on a page, and updates the threshold entry on the corresponding SPAM page to reflect the new space utilization of the data page. The data page and SPAM pages are not flushed to disk.
2. Another process notifies the first process that it would like to access the SPAM page being held by the process. The first process flushes the SPAM page changes to disk and releases the page. Note that it has not flushed the data page.
3. The first process then terminates abnormally (for example, from the DCL STOP/IDENTIFICATION command). Since that process never flushed the data page to disk, it never wrote the changes to the Recovery Unit Journal (RUJ) file. Since there were no changes in the RUJ file for that data page then the Database Recovery (DBR) process did not need to roll back any changes to the page. The SPAM page retains the threshold update change made above even though the data page was never flushed to disk.

While it would be possible to create mechanisms to ensure that SPAM pages do not become out of synch with their corresponding data pages, the performance impact would not be trivial. Since these errors are relatively rare and the impact is not significant, then the introduction of these errors is considered to be part of the normal operation of Oracle Rdb. If it can be proven that the errors are not due to the scenario above, then Oracle Product Support should be contacted.

PGSPAMENT and PGSPMCLST errors may be corrected by doing any one of the following operations:

- Recreate the database by performing:
    1. SQL EXPORT
    2. SQL DROP DATABASE
    3. SQL IMPORT
- Recreate the database by performing:
    1. RMU/BACKUP
    2. SQL DROP DATABASE

3. RMU/RESTORE

- Repair the SPAM pages by using the RMU/REPAIR command. Note that the RMU/REPAIR command does not write its changes to an after−image journal (AIJ) file. Therefore, Oracle recommends that a full database backup be performed immediately after using the RMU/REPAIR command.

8.4 Known Problems and Restrictions in All Interfaces for Release 7.0 and Earlier

The following problems and restrictions from release 7.0 and earlier still exist.

8.4.1 Converting Single−File Databases

Because of a substantial increase in the database root file information for V7.0, you should ensure that you have adequate disk space before you use the RMU Convert command with single−file databases and V7.0 or higher.

The size of the database root file of any given database increases a maximum of about 600 disk blocks. The actual increase depends mostly on the maximum number of users specified for the database.

8.4.2 Row Caches and Exclusive Access

If a table has a row−level cache defined for it, the Row Cache Server (RCS) may acquire a shared lock on the table and prevent any other user from acquiring a Protective or Exclusive lock on that table.

8.4.3 Exclusive Access Transactions May Deadlock with RCS Process

If a table is frequently accessed by long running transactions that request READ/WRITE access reserving the table for EXCLUSIVE WRITE and if the table has one or more indexes, you may experience deadlocks between the user process and the Row Cache Server (RCS) process.

There are at least three suggested workarounds to this problem:

- Reserve the table for SHARED WRITE
- Close the database and disable row cache for the duration of the exclusive transaction
- Change the checkpoint interval for the RCS process to a time longer than the time required to complete the batch job and then trigger a checkpoint just before the batch job starts. Set the interval back to a smaller interval after the checkpoint completes.

8.4.4 Strict Partitioning May Scan Extra Partitions

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

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

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

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

8.4.5 Restriction When Adding Storage Areas with Users Attached to Database

If you try to interactively add a new storage area where the page size is less than the smallest existing page size and the database has been manually opened or users are active, the add operation fails with the following errors:

```
%RDMS−F−NOEUACCESS, unable to acquire exclusive access to database
```

or

```
%RDB−F−SYS_REQUEST, error from system services request
−RDMS−F−FILACCERR, error opening database root DKA0:[RDB]TEST.RDB;1
−SYSTEM−W−ACCONFLICT, file access conflict
```

You can make this change only when no users are attached to the database and, if the database is set to OPEN IS MANUAL, the database is closed. Several internal Oracle Rdb data structures are based on the minimum page size and these structures cannot be resized if users are attached to the database.

Furthermore, because this particular change is not recorded in the AIJ, any recovery scenario fails. Note also that if you use .aij files, you must backup the database and restart after−image journaling because this change invalidates the current AIJ recovery.

8.4.6 Multiblock Page Writes May Require Restore Operation

If a node fails while a multiblock page is being written to disk, the page in the disk becomes inconsistent, and is detected immediately during failover. (Failover is the recovery of an application by restarting it on another computer.) The problem is rare, and occurs because only single−block I/O operations are guaranteed by OpenVMS to be written atomically. This problem has never been reported by any customer and was detected only during stress tests in our labs.

Correct the page by an area−level restore operation. Database integrity is not compromised, but the affected area is not available until the restore operation completes.

A future release of Oracle Rdb will provide a solution that guarantees multiblock atomic write operations. Cluster failovers will automatically cause the recovery of multiblock pages, and no manual intervention will be required.

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

When a group of copy processes initiated by the Replication Option (formerly Data Distributor) begins running after an application has begun modifying the database, the copy processes catch up to the application and are not able to process database pages that are logically ahead of the application in the RDB$CHANGES system relation. The copy processes all align waiting for the same database page and do not move on until the application has released it. The performance of each copy process degrades because it is being paced by the application.

When a copy process completes updates to its respective remote database, it updates the RDB$TRANSFERS system relation and then tries to delete any RDB$CHANGES rows not needed by any transfers. During this process, the RDB$CHANGES table cannot be updated by any application process, holding up any database updates until the deletion process is complete. The application stalls while waiting for the RDB$CHANGES table. The resulting contention for RDB$CHANGES SPAM pages and data pages severely impacts performance throughput, requiring user intervention with normal processing.

This is a known restriction in V4.0 and higher. Oracle Rdb uses page locks as latches. These latches are held only for the duration of an action on the page and not to the end of transaction. The page locks also have blocking asynchronous system traps (ASTs) associated with them. Therefore, whenever a process requests a page lock, the process holding that page lock is sent a blocking AST (BLAST) by OpenVMS. The process that receives such a blocking AST queues the fact that the page lock should be released as soon as possible. However, the page lock cannot be released immediately.

Such work requests to release page locks are handled at verb commit time. An Oracle Rdb verb is an Oracle Rdb query that executes atomically, within a transaction. Therefore, verbs that require the scan of a large table, for example, can be quite long. An updating application does not release page locks until its verb has completed.

The reasons for holding on to the page locks until the end of the verb are fundamental to the database management system.

8.5 SQL Known Problems and Restrictions for Oracle Rdb Release 7.0 and Earlier

The following problems and restrictions from Oracle Rdb Release 7.0 and earlier still exist.

8.5.1 ARITH_EXCEPT or Incorrect Results Using LIKE IGNORE CASE

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

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

8.5.2 Different Methods of Limiting Returned Rows from Queries

You can establish the query governor for rows returned from a query by using either the SQL SET QUERY LIMIT statement or a logical name. This note describes the differences between the two mechanisms.

If you define the RDMS$BIND_QG_REC_LIMIT logical name to a small value, the query often fails with no rows returned regardless of the value assigned to the logical. The following example demonstrates setting the limit to 10 rows and the resulting failure:

```
$ DEFINE RDMS$BIND_QG_REC_LIMIT 10
$ SQL$
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES;
%RDB−F−EXQUOTA, Oracle Rdb runtime quota exceeded
−RDMS−E−MAXRECLIM, query governor maximum limit of rows has been reached
```

Interactive SQL must load its metadata cache for the table before it can process the SELECT statement. In this example, interactive SQL loads its metadata cache to allow it to check that the column EMPLOYEE_ID really exists for the table. The queries on the Oracle Rdb system relations RDB$RELATIONS and RDB$RELATION_FIELDS exceed the limit of rows.

Oracle Rdb does not prepare the SELECT statement, let alone execute it. Raising the limit to a number less than 100 (the cardinality of EMPLOYEES) but more than the number of columns in EMPLOYEES (that is, the number of rows to read from the RDB$RELATION_FIELDS system relation) is sufficient to read each column definition.

To see an indication of the queries executed against the system relations, define the RDMS$DEBUG_FLAGS logical name as "S" or "B".

If you set the row limit using the SQL SET QUERY statement and run the same query, it returns the number of rows specified by the SQL SET QUERY statement before failing:

```
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SET QUERY LIMIT ROWS 10;
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES;
EMPLOYEE_ID
00164
00165
   .
   .
   .
00173
%RDB-E-EXQUOTA, Oracle Rdb runtime quota exceeded
-RDMS-E-MAXRECLIM, query governor maximum limit of rows has been reached
```

The SET QUERY LIMIT specifies that only user queries be limited to 10 rows. Therefore, the queries used to load the metadata cache are not restricted in any way.

Like the SET QUERY LIMIT statement, the SQL precompiler and module processor command line qualifiers (QUERY_MAX_ROWS and SQLOPTIONS=QUERY_MAX_ROWS) only limit user queries.

Keep the differences in mind when limiting returned rows using the logical name RDMS$BIND_QG_REC_LIMIT. They may limit more queries than are obvious. This is important when using 4GL tools, the SQL precompiler, the SQL module processor, and other interfaces that read the Oracle Rdb system relations as part of query processing.

8.5.3 Suggestions for Optimal Use of SHARED DATA DEFINITION Clause for Parallel Index Creation

The CREATE INDEX process involves the following steps:

1. Process the metadata.
2. Lock the index name.
   Because new metadata (which includes the index name) is not written to disk until the end of the index process, Oracle Rdb must ensure index name uniqueness across the database during this time by taking a special lock on the provided index name.
3. Read the table for sorting by selected index columns and ordering.
4. Sort the key data.
5. Build the index (includes partitioning across storage areas).
6. Write new metadata to disk.

Step 6 is the point of conflict with other index definers because the system relation and indexes are locked like any other updated table.

Multiple users can create indexes on the same table by using the RESERVING table_name FOR SHARED DATA DEFINITION clause of the SET TRANSACTION statement. For optimal usage of this capability, Oracle Rdb suggests the following guidelines:

- You should commit the transaction immediately after the CREATE INDEX statement so that locks on the table are released. This avoids lock conflicts with other index definers and improves overall concurrency.
- By assigning the location of the temporary sort work files SORTWORK0, SORTWORK1, ... , SORTWORK9 to different disks for each parallel process that issues the SHARED DATA DEFINITION statement, you can increase the efficiency of sort operations. This minimizes

any possible disk I/O bottlenecks and allows overlap of the SORT read/write cycle.

- If possible, enable global buffers and specify a buffer number large enough to hold a sufficient amount of table data. However, do not define global buffers larger than the available system physical memory. Global buffers allow sharing of database pages and thus result in disk I/O savings. That is, pages are read from disk by one of the processes and then shared by the other index definers for the same table, reducing the I/O load on the table.
- If global buffers are not used, ensure that enough local buffers exist to keep much of the index cached (use the RDM$BIND_BUFFERS logical name or the NUMBER OF BUFFERS IS clause in SQL to change the number of buffers).
- To distribute the disk I/O load, store the storage areas for the indexes on separate disk drives. Note that using the same storage area for multiple indexes results in contention during the index creation (Step 5) for SPAM pages.
- Consider placing the .ruj file for each parallel definer on its own disk or an infrequently used disk.
- Even though snapshot I/O should be minimal, consider disabling snapshots during parallel index creation.
- Refer to the Oracle Rdb7 Guide to Database Performance and Tuning to determine the appropriate working set values for each process to minimize excessive paging activity. In particular, avoid using working set parameters where the difference between WSQUOTA and WSEXTENT is large. The SORT utility uses the difference between these two values to allocate scratch virtual memory. A large difference (that is, the requested virtual memory grossly exceeds the available physical memory) may lead to excessive page faulting.
- The performance benefits of using SHARED DATA DEFINITION can best be observed when creating many indexes in parallel. The benefit is in the average elapsed time, not in CPU or I/O usage. For example, when two indexes are created in parallel using the SHARED DATA DEFINITION clause, the database must be attached twice, and the two attaches each use separate system resources.
- Using the SHARED DATA DEFINITION clause on a single−file database or for indexes defined in the RDB$SYSTEM storage area is not recommended.

The following table displays the elapsed time benefit when creating multiple indexes in parallel with the SHARED DATA DEFINITION clause. The table shows the elapsed time for ten parallel process index creations (Index1, Index2, ... Index10) and one process with ten sequential index creations (All10). In this example, global buffers are enabled and the number of buffers is 500. The longest time for a parallel index creation is Index7 with an elapsed time of 00:02:34.64, compared to creating ten indexes sequentially with an elapsed time of 00:03:26.66. The longest single parallel create index elapsed time is shorter than the elapsed time of creating all ten of the indexes serially.

Table 8−2 Elapsed Time for Index Creations

Index Create Job	Elapsed Time
Index1	00:02:22.50
Index2	00:01:57.94
Index3	00:02:06.27
Index4	00:01:34.53
Index5	00:01:51.96
Index6	00:01:27.57
Index7	00:02:34.64
Index8	00:01:40.56

Index9	00:01:34.43
Index10	00:01:47.44
All10	00:03:26.66

8.5.4 Side Effect When Calling Stored Routines

When calling a stored routine, you must not use the same routine to calculate argument values by a stored function. For example, if the routine being called is also called by a stored function during the calculation of an argument value, passed arguments to the routine may be incorrect.

The following example shows a stored procedure P being called during the calculation of the arguments for another invocation of the stored procedure P:

```
SQL> create module M
cont>     language SQL
cont>
cont>     procedure P (in :a integer, in :b integer, out :c integer);
cont>     begin
cont>     set :c = :a + :b;
cont>     end;
cont>
cont>     function F () returns integer
cont>     comment is 'expect F to always return 2';
cont>     begin
cont>     declare :b integer;
cont>     call P (1, 1, :b);
cont>     trace 'returning ', :b;
cont>     return :b;
cont>     end;
cont> end module;
SQL>
SQL> set flags 'TRACE';
SQL> begin
cont> declare :cc integer;
cont> call P (2, F(), :cc);
cont> trace 'Expected 4, got ', :cc;
cont> end;
~Xt: returning 2
~Xt: Expected 4, got 3
```

The result as shown above is incorrect. The routine argument values are written to the called routine's parameter area before complex expression values are calculated. These calculations may (as in the example) overwrite previously copied data.

The workaround is to assign the argument expression (in this example calling the stored function F) to a temporary variable and pass this variable as the input for the routine. The following example shows the workaround:

```
SQL> begin
cont> declare :bb, :cc integer;
cont> set :bb = F();
cont> call P (2, :bb, :cc);
cont> trace 'Expected 4, got ', :cc;
cont> end;
~Xt: returning 2
```

```
~Xt: Expected 4, got 4
```

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

8.5.5 Considerations When Using Holdable Cursors

If your applications use holdable cursors, be aware that after a COMMIT or ROLLBACK statement is executed, the result set selected by the cursor may not remain stable. That is, rows may be inserted, updated, and deleted by other users because no locks are held on the rows selected by the holdable cursor after a commit or rollback occurs. Moreover, depending on the access strategy, rows not yet fetched may change before Oracle Rdb actually fetches them.

As a result, you may see the following anomalies when using holdable cursors in a concurrent user environment:

- If the access strategy forces Oracle Rdb to take a data snapshot, the data read and cached may be stale by the time the cursor fetches the data.
  For example, user 1 opens a cursor and commits the transaction. User 2 deletes rows read by user 1 (this is possible because the read locks are released). It is possible for user 1 to report data now deleted and committed.
- If the access strategy uses indexes that allow duplicates, updates to the duplicates chain may cause rows to be skipped, or even revisited.
  Oracle Rdb keeps track of the dbkey in the duplicate chain pointing to the data that was fetched. However, the duplicates chain could be revised by the time Oracle Rdb returns to using it.

Holdable cursors are a very powerful feature for read−only or predominantly read−only environments. However, in concurrent update environments, the instability of the cursor may not be acceptable. The stability of holdable cursors for update environments will be addressed in future versions of Oracle Rdb.

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

8.5.6 AIJSERVER Privileges

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

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

- ALTPRI − This privilege allows the AIJSERVER to adjust its own priority to ensure adequate quorum (CPU utilization) to prompt message processing.
- PSWAPM − This privilege allows the AIJSERVER to enable and disable process swapping, also necessary to ensure prompt message processing.

- SETPRV − This privilege allows the AIJSERVER to temporarily set any additional privileges it may need to access the standby database or its server processes.
- SYSPRV − This privilege allows the AIJSERVER to access the standby database rootfile, if necessary.
- WORLD − This privilege allows the AIJSERVER to more accurately detect standby database server process failure and handle network failure more reliably.

| Contents