Oracle® Rdb for OpenVMS

Table of Contents

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 7.3.1.2

October 2014

Oracle Rdb Release Notes, Release 7.3.1.2 for OpenVMS

Primary Author: Rdb Engineering and Documentation group

Contents

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb Release 7.3.1.2. 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 7.3.1.2.

Access to Oracle Support

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

Document Structure

This manual consists of the following chapters:

Chapter 1	Describes how to install Oracle Rdb Release 7.3.1.2.
Chapter 2	Describes problems corrected in Oracle Rdb Release 7.3.1.2.
Chapter 3	Describes problems corrected in Oracle Rdb Release 7.3.1.1.
Chapter 4	Describes problems corrected in Oracle Rdb Release 7.3.1.0.
Chapter 5	Describes enhancements introduced in Oracle Rdb Release 7.3.1.2.
Chapter 6	Describes enhancements introduced in Oracle Rdb Release 7.3.1.1.
Chapter 7	Describes enhancements introduced in Oracle Rdb Release 7.3.1.0.
Chapter 8	Provides information not currently available in the Oracle Rdb documentation set.
Chapter 9	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb Release 7.3.1.2.

Chapter 1
Installing Oracle Rdb Release 7.3.1.2

This software update is installed using the OpenVMS VMSINSTAL utility.

NOTE

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

1.1 Oracle Rdb on 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 for one platform is available on the other platform. However, certain differences between the platforms may result in minor capability and functionality differences.

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

1.2 Requirements

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

- This Oracle Rdb release requires the following OpenVMS environments:
    - ♦ OpenVMS Alpha V8.3 to V8.4−x.
    - ♦ OpenVMS Industry Standard 64 V8.3 to V8.4−x.
- Oracle Rdb must be shutdown before you install this update kit. That is, the command file SYS$STARTUP:RMONSTOP73.COM should be executed before proceeding with this installation. If you have an OpenVMS cluster, you must shutdown the Rdb Release 7.3 monitor on all nodes in the cluster before proceeding.
- After executing RMONSTOP73.COM, no process on any system in the cluster should have any existing RDMSHRP73.EXE image activated. See Section 1.2.1 for additional information.
- The installation requires approximately 280,000 blocks for OpenVMS Alpha systems.
- The installation requires approximately 500,000 blocks for OpenVMS I64 systems.
- The following OpenVMS Mandatory Update from HP needs to be installed on Itanium 8.4 systems before installing this kit: VMS84I_MUP−V0500. A reboot is required after the MUP is installed. The problem description for this fix is: The OpenVMS OTS library string comparison routines OTS$STRCMP_LSSP and OTS$STRCMP_LEQP might return inaccurate results when used with specific string patterns.
  This Mandatory Kit has the following dependencies:
    - ♦ VMS84I_SYS−V0300
    - ♦ VMS84I_UPDATE−V0800
    - ♦ VMS84I_PCSI−V0400
- The following OpenVMS patches should be installed before the Rdb kit is installed (this corrects a problem that caused processes to go into an RWINS state, thus prompting a reboot of the system to clear the error):
    - ♦ VMS84A_SYS−V0500 (for Alpha)
      This kit supersedes VMS84A_SYS−V0400. The following remedial kit(s) must be installed BEFORE installation of this kit:
        - ◊ VMS84A_PCSI−V0400
        - ◊ VMS84A_UPDATE−V0900
    - ♦ VMS84I_SYS−V0500 (for Itanium)
      This kit supersedes VMS84I_SYS−V0400. The following remedial kit(s) must be installed BEFORE installation of this kit:
        - ◊ VMS84I_PCSI−V0400
        - ◊ VMS84I_UPDATE−V0900

  Please contact your HP support representative if you have questions or need more information about these updates.
- Oracle strongly recommends that all available OpenVMS patches are installed on all systems prior to installing Oracle Rdb. Contact your HP support representative for more information and assistance.

1.2.1 Ensure No Processes Have RDMSHRP Image Activated

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

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

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

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

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

OpenVMS system analyzer

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

$ STOP/IDENTIFICATION=2729F16D
```

1.3 Intel Itanium Processor 9300 "Tukwila" Support

For this release of Oracle Rdb on HP Integrity servers, the Intel Itanium Processor 9300 series, code named "Tukwila", is the newest processor supported.

1.4 Maximum OpenVMS Version Check

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

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

1.5 Database Format Changed

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

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

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

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

1.6 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 V7.3 format. The RMU Convert command for Oracle Rdb V7.3 supports conversions from Oracle Rdb V7.0, V7.1 and V7.2 format databases only. If you have an Oracle Rdb V3.0 through V6.1 format database or database backup, you must convert it to at least Oracle Rdb V7.0 format and then convert it to Oracle Rdb V7.3 format. For example, if you have a V4.2 format database, you must convert it first to at least Oracle Rdb V7.0 format, then convert it to Oracle Rdb V7.3 format.

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

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

  ```
  @SYS$UPDATE:VMSINSTAL RDBV73120IM073 device−name
  ```
- To install the Oracle Rdb for OpenVMS Alpha kit

  ```
  @SYS$UPDATE:VMSINSTAL RDBV73121AM073 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, "RDBVMSV7.3".

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

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

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

   ```
   $ LIBRARY /TEXT /EXTRACT=RDBVMSV7.3 −
   _$ /OUT=RDBVMS.EPC$DEF SYS$SHARE:EPC$FACILITY.TLB
   ```
2. Insert the facility definition into the Oracle TRACE administration database.

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

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

1.10 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 7.3 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.

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

When moving applications from existing Alpha or VAX configurations to new environments containing Integrity Server systems, there are numerous possible paths depending on the requirements of individual sites. In general, this can be as straightforward as adding a new node to an already existing AlphaServer systems cluster or standalone system, except the node is an 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 7.3 on Integrity and Alpha nodes. 4. Convert database(s) to the Rdb 7.3 structure level using RMU/CONVERT. 5. Verify database(s) again using RMU/VERIFY/ALL. 6. Backup database(s) using RMU/BACKUP. 7. Access database(s) from Alpha and Integrity directly by specifying database root file

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

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

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

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

Chapter 2
Software Errors Fixed in Oracle Rdb Release 7.3.1.2

This chapter describes software errors that are fixed by Oracle Rdb Release 7.3.1.2.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 Ranked Index Bugchecks − PSII2REMOVEDUPBBC & PSII2INSERTDUPBBC

Bug 17383599

In prior versions of Oracle Rdb, applications trying to insert or remove records using a ranked index may bugcheck with an exception similar to the following:

```
Exception at 00000000xxxxxxxx : RDMSHRP72\PSII2REMOVEDUPBBC + 0000xxxx
%COSI−F−BUGCHECK, internal consistency failure
```

A bugcheck may also be raised with the following exception:

```
Exception at 00000000xxxxxxxx : RDMSHRP72\PSII2INSERTDUPBBC + 0000xxxx
%COSI−F−BUGCHECK, internal consistency failure
```

In addition, the following exception may be raised during record retrieval:

```
%RDMS−F−NODBK, 61:117:29 does not point to a data record
```

This problem may also lead to a corruption of the index on−disk: an RMU VERIFY INDEX of the database will indicate that the index is corrupt with an exception similar to:

```
RMU−E−BADDBKFET, Error fetching dbkey 61:117:29
```

Depending on the transaction mix of insertions and deletions, it is possible that these problems may not cause any on−disk index corruption: the problem may only be a transient error, affecting in−memory structures only.

These problems are more likely to occur when there are a large number of duplicates within the ranked index entries.

A possible workaround is to rebuild the index if RMU/VERIFY has shown an exception similar to the one cited above. The rebuild of the index will rectify any ranked index corruptions. However, in prior versions of Oracle Rdb, the same problem may re−occur on subsequent transactions.

These problems were actually corrected in Oracle Rdb Release 7.3.1.1.

2.1.2 Wrong Result From a Nested UNION Query With OR Predicate

Bug 18825653

In prior releases of Oracle Rdb, a complex query referencing a view containing a UNION may return the wrong result when an OR predicate is involved in the outer query.

For example, the following query is expected to return some rows and it does not.

```
SELECT C1, C3, C6 FROM Q_VIEW
WHERE C6='20' AND (C3= 'KMT' or C3= 'HFM');
0 rows selected
```

However, if the OR predicate is omitted, the query returns the rows correctly.

```
SELECT C1, C3, C6 FROM Q_VIEW
WHERE C6='20' AND C3= 'KMT';
 C1          C3         C6
 EGP         KMT        20
1 row selected
```

The views are defined as follows:

```
create view Q_VIEW (QC1, QC2, QC3, QC4, QC5, QC6)
as
select (CI.C1, CI.C2, CI.C3, CI.C4, CI.C5, CI.C6)
from CI_VIEW CI,        ! 1 row
     TT1                ! empty table
UNION
select (CI.C1, CI.C2, CI.C3, CI.C4, CI.C5, CI.C6)
from CI_VIEW CI inner join      ! 1 row
     TT2 on (TT2.C1 = CI.C1)    ! 1 row
;

create view CI_VIEW (CI1, CI2, CI3, CI4, CI5, CI6)
as
select (T1.C1, T1.C2, T1.C3, T1.C4, T1.C5,
        CAST('CGS' AS CHAR(3)   ! C6
        )
from T1 inner join T2           ! empty tables
     on (T1.C1 = T2.C1)
UNION
select (T3.C1, T3.C2, T3.C3, T3.C4, T5.C5,
        CASE WHEN EXISTS (SELECT * FROM T6
                          WHERE T6.C3= T4.C3)
                 THEN CAST('EGP' AS CHAR(3))
                 ELSE CAST('   ' AS CHAR(3))
            END                     ! C6
from T3 inner join                      ! 1 row
     T4 on (T4.C3 = T3.C3) inner join   ! 1 row
     T5 on (T5.C1 = T3.C1)              ! 1 row
where T3.C7 = 'XXXXX';
```

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

1. The main query selects from a nested UNION view with an OR predicate in the WHERE clause.
2. The main view, Q_VIEW, is a UNION query of two legs where another view, CI_VIEW, is nested within the join.
3. CI_VIEW is also a UNION query where each leg joins another table.
4. The first UNION leg joins an empty table but the second one joins three main tables using inner join. The column C6 of the select clause contains a CASE statement with subselect joining another table.

Currently there is no workaround for this problem.

These problems have been corrected in Oracle Rdb Release 7.3.1.2.

2.1.3 Dialect SQL99 Does Not Use Max Key Lookup

Bug 18517510

In Oracle Rdb Release 7.3.1.1, the optimizer avoids using "Max key lookup" and "Min key lookup" optimizations if any of these dialects are used: SQL92, SQL99, SQL2011, ORACLE LEVEL1, ORACLE LEVEL2 or ORACLE LEVEL3.

Many customers have queries where the performance benefits of the "Max key lookup" strategy outweigh the theoretical correctness of getting NULL elimination warnings.

For example, the following query applies "Max key lookup" in a MF_PERSONNEL database with SALARY_HISTORY_TEST_IDX defined as sorted ranked but does not use that strategy when using (for example) SET DIALECT 'SQL99'.

```
SQL> CREATE INDEX SALARY_HISTORY_TEST_IDX
cont> ON SALARY_HISTORY (EMPLOYEE_ID, SALARY_START)
cont> TYPE IS SORTED RANKED;
SQL>
SQL> SELECT MAX(SALARY_START)
cont> FROM SALARY_HISTORY
cont> WHERE EMPLOYEE_ID = '00374';
Tables:
  0 = SALARY_HISTORY
Aggregate: 0:MAX (0.SALARY_START)
Index only retrieval of relation 0:SALARY_HISTORY
  Index name  SALARY_HISTORY_TEST_IDX [1:1]      Max key lookup
    Keys: 0.EMPLOYEE_ID = '00374'

 10-OCT-1982 00:00:00.00
1 row selected
SQL> set dialect 'sql99' ;
SQL> SELECT MAX(SALARY_START)
cont> FROM SALARY_HISTORY
cont> WHERE EMPLOYEE_ID = '00374';
Tables:
  0 = SALARY_HISTORY
Aggregate: 0:MAX (0.SALARY_START)
Index only retrieval of relation 0:SALARY_HISTORY
  Index name  SALARY_HISTORY_TEST_IDX [1:1]
    Keys: 0.EMPLOYEE_ID = '00374'

 10-OCT-1982 00:00:00.00
1 row selected
```

There is no workaround for this problem except to avoid the listed dialects when using MIN and MAX optimizations.

These problems have been corrected in Oracle Rdb Release 7.3.1.2.

2.1.4 Inner Join Query With NOT LIKE Predicate Slows Down

Bug 18081992

In prior releases of Oracle Rdb, an inner join query using a predicate with a NOT operator may slow down when using Cross strategy. Note that in this example, the NOT LIKE operation is transformed to a NOT STARTING WITH operation.

The problem is caused by the NOT operator during the costing as a Cross join. With the fix, the strategy is switched to a Match join by disabling the transitivity in such cases.

For example, the following query may run for an extended time.

```
SELECT count(*)
FROM (
       SELECT
               T1.DEV_ID,
               T1.OPER_ID,
               T1.OPER_SEQ_NO,
               T1.STEP_ID
       FROM T1
       WHERE (((T1.STEP_ID)<>'------'))
      ) stp
    INNER JOIN T2
      ON stp.OPER_ID = T2.OPER_ID
    where
        stp.OPER_ID Not Like 'WC%' and
        T2.STEP_ID Like 'PP%'
    ;
Tables:
  0 = T1
  1 = T2
Aggregate: 0:COUNT (*) Q2
Cross block of 2 entries  Q2
  Cross block entry 1
    Leaf#01 NdxOnly 1:T2 Card=123039
      Bool: 1.STEP_ID STARTING WITH 'PP'
      FgrNdx  T2_NDX1 [0:0] Fan=12
      BgrNdx1 T2_NDX2 [1:1] Fan=15
        Keys: 1.STEP_ID STARTING WITH 'PP'
  Cross block entry 2
    Conjunct: 0.OPER_ID = 1.OPER_ID
    Merge of 1 entries  Q2
      Merge block entry 1  Q3
      Conjunct: NOT (0.OPER_ID STARTING WITH 'WC')
      Leaf#02 BgrOnly 0:T1 Card=158894
        Bool: 0.STEP_ID <> '------'
        BgrNdx1 T1_IDX [0:0] Fan=11
          Bool: (NOT (0.OPER_ID STARTING WITH 'WC')) AND (NOT (0.OPER_ID
                 STARTING WITH 'WC')) AND (0.STEP_ID <> '------')
```

The query runs fast if transitivity is disabled, which uses a Match strategy. When executing the same query as before, you can see the altered strategy which leads to a reduced execution time.

```
Tables:
  0 = T1
  1 = T2
Aggregate: 0:COUNT (*) Q2
Conjunct: 0.OPER_ID = 1.OPER_ID
Match  Q2
  Outer loop
  Match_Key:0.OPER_ID
    Sort: 0.OPER_ID(a)
    Merge of 1 entries  Q2
      Merge block entry 1  Q3
      Conjunct: NOT (0.OPER_ID STARTING WITH 'WC')
      Leaf#01 BgrOnly 0:T1 Card=158894
        Bool: 0.STEP_ID <> '------'
        BgrNdx1 T1_IDX [0:0] Fan=11
          Bool: (NOT (0.OPER_ID STARTING WITH 'WC')) AND (0.STEP_ID <>
                '------')
  Inner loop      (zig-zag)
  Match_Key:1.OPER_ID
  Index_Key:OPER_ID, STEP_ID
    Conjunct: 1.STEP_ID STARTING WITH 'PP'
    Index only retrieval of relation 1:T2
      Index name  T2_IDX1 [0:0]
        Bool: 1.STEP_ID STARTING WITH 'PP'

      172849
1 row selected
```

A workaround may be to disable transitivity using the SET FLAGS 'NOTRANSITIVITY' statement.

This problem has been corrected in Oracle Rdb Release 7.3.1.2.

2.1.5 Unable to Call External Routine When Attached Remotely to a Database

Bug 13640883

When an external routine which uses SQL statements, attached to the same database that was remotely accessed, then it would fail with an error −RDB−E−INVALID_SEC_INF, invalid security information. For example:

```
SQL> attach 'filename remhst"username password"::mydb';
SQL> declare :syu, :ssu, :cru rdb$object_name = '';
SQL> declare :sc int = 0;
SQL> begin
cont> call show_user (:sc, :syu, :ssu, :cru);
cont> trace 'sqlcode -> ', :sc;
cont> if :sc < 0
cont> then
cont>     call sql_signal ();
cont> end if;
cont> end;
%RDB-E-EXTFUN_FAIL, external routine failed to compile or execute successfully
-RDMS-E-RTN_ERROR, routine "SQL_SIGNAL" generated an error during execution
-RDB-E-INVALID_SEC_INF, invalid security information
SQL>
```

A workaround would be to include explicit USER/USING clauses in the external routine to attach to the local database.

This problem has been corrected in Oracle Rdb Release 7.3.1.2.

2.1.6 RDMDBRBUG Bugcheck at RUJUTL$BIJBL_GET_FORWARD + 1E0

Bug 18506440

In rare circumstances, a Database Recovery Process (DBR) may fail an internal sanity check and bugcheck with the above exception (the actual offset would depend on the Oracle Rdb version and OpenVMS platform). After the recovery fails, the database will shutdown all users. However, the next database attach will cause another DBR process to start. This follow−up DBR should be able to successfully complete the original recovery, allowing access to the database once more.

The problem was caused by mismanagement of a data buffer while reading the Run Unit Journal (RUJ) recovery file. As such, there is no user workaround. There is no database or RUJ file corruption associated with this failure.

This problem has been corrected in Oracle Rdb Release 7.3.1.2.

2.1.7 Join Query Returns Wrong Result With Index Counts Lookup Using Ranked Index

Bug 19595377

In Oracle Rdb Release 7.3.1.1, a simple query joining another query of "select count(*)" with "Index counts lookup" using a sorted ranked index could return the wrong result.

```
!
! The following query returns correct result using sorted index:
!
select e.employee_id,
  (select count(*) from salary_history s where e.employee_id=s.employee_id)
  from employees e limit to 5 rows;
Tables:
  0 = EMPLOYEES
  1 = SALARY_HISTORY
Cross block of 2 entries  Q0
  Cross block entry 1
    Firstn: 5
    Index only retrieval of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Cross block entry 2
    Aggregate: 0:COUNT (*) Q2
    Index only retrieval of relation 1:SALARY_HISTORY
      Index name  SH_EMPLOYEE_ID [1:1]
        Keys: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
 EMPLOYEE_ID
 00164                          4
 00165                         12
 00166                          6
 00167                          5
 00168                         11
5 rows selected
```

```
! now drop the sorted index and create a ranked index
!
drop index sh_employee_id;
create index sh_employee_id on salary_history (employee_id) type is sorted
ranked;

! If using sorted ranked index, the query returns wrong result
!
select e.employee_id,
  (select count(*) from salary_history s where e.employee_id=s.employee_id)
  from employees e limit to 5 rows;
Tables:
  0 = EMPLOYEES
  1 = SALARY_HISTORY
Cross block of 2 entries  Q0
  Cross block entry 1
    Firstn: 5
    Index only retrieval of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Cross block entry 2
    Aggregate: 0:COUNT (*) Q2
    Index only retrieval of relation 1:SALARY_HISTORY
      Index name  SH_EMPLOYEE_ID [1:1]       Index counts lookup
        Keys: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
 EMPLOYEE_ID
 00164                          0
 00165                          0
 00166                          0
 00167                          0
 00168                          0
5 rows selected
```

There is no workaround for this problem except to avoid using sorted ranked indices.

This problem has been corrected in Oracle Rdb Release 7.3.1.2.

2.1.8 Unexpected Zero Cardinality Set for Some Tables

Bug 19611808

In prior releases of Oracle Rdb V7.3, it was possible that a sequential scan of a table following an INSERT or DELETE statement might erroneously set the table cardinality to zero.

In turn, this change could cause poor query performamce when the optimizer used sequential access to query that which it thought was an empty table.

This problem has been corrected in Oracle Rdb Release 7.3.1.2. Oracle recommends that all customers execute RMU Collect after this update is installed.

```
$ RMU/COLLECT OPTIMIZER_STATISTICS/Statistics=Cardinality/NoIndexes dbname
```

2.1.9 Query With Aggregate Bugchecks Using Match Strategy

Bug 19622034

In prior releases of Oracle Rdb, the optimizer may bugcheck while processing an aggregate subquery such as EXISTS in the following example.

```
SELECT count(*)
FROM t1, t2, t3, t4
WHERE
 t1.col=t4.col and
 t2.col=t1.col and
 t1.status='E' and
 t1.col=t3.col and
 EXISTS (SELECT * FROM t5
    WHERE t5.col=t1.col) and
 t2.fdate = (SELECT max(fdate)
              FROM t2 t5
              WHERE t5.col=t2.col
                and fdate<='20140912');
%RDMS−I−BUGCHKDMP, generating bugcheck dump file DISK:[USERNAME]RDSBUGCHK.DMP;
```

A workaround may be to use a query outline to use a CROSS JOIN instead of a MATCH JOIN, or to replace the EXISTS clause with a (SELECT COUNT(*) FROM ... WHERE) > 0.

```
SELECT count(*)
FROM t1, t2, t3, t4
WHERE
 t1.col=t4.col and
 t2.col=t1.col and
 t1.status='E' and
 t1.col=t3.col and
 (SELECT COUNT(*) FROM t5
    WHERE t5.col=t1.col) > 0 and
 t2.fdate = (SELECT max(fdate)
              FROM t2 t5
              WHERE t5.col=t2.col
                and fdate<='20140912');
```

This problem has been corrected in Oracle Rdb Release 7.3.1.2.

2.2 SQL Errors Fixed

2.2.1 Unexpected Zero Result From COUNT Aggregate

Bug 17908875

In prior releases of Oracle Rdb, it was possible, in rare cases, for the Rdb optimizer to use a dynamic retrieval strategy to solve a COUNT aggregate using more than one SORTED RANKED background index. Depending on the index chosen by the estimation (Estim) phase as the most likely index, it was possible for the COUNT to erroneously return zero when the "Index counts lookup" optimization was used.

The following example shows such a strategy.

```
SQL> SELECT COUNT(*)
cont> FROM SAMPLE
cont> WHERE SAMPLE_BX = 7001
cont> AND SAMPLE_AC = 'Xxxx'
cont> AND SAMPLE_XC = 'Yyyy'
cont> AND SAMPLE_CD = '1234567'
cont> ;
~S#0002
Tables:
  0 = SAMPLE
Aggregate: 0:COUNT (*) Q2
Leaf#01 BgrOnly 0:SAMPLE Card=1492231
  Bool: (0.SAMPLE_BX = 7001) AND (0.SAMPLE_AC = 'Xxxx') AND (
        0.SAMPLE_XC = 'Yyyy') AND (0.SAMPLE_CD = '1234567')
  BgrNdx1 SAMPLE_KEY [3:3] Fan=1
    Keys: (0.SAMPLE_BX = 7001) AND (0.SAMPLE_AC = 'Xxxx') AND (
          0.SAMPLE_CD = '1234567')
  BgrNdx2 SAMPLE_RANKED_1 [4:4] Fan=51  Index counts lookup
    Keys: (0.SAMPLE_CD = '1234567') AND (0.SAMPLE_BX = 7001) AND (
          0.SAMPLE_XC = 'Yyyy') AND (0.SAMPLE_AC = 'Xxxx')
  BgrNdx3 SAMPLE_RANKED_2 [4:4] Fan=51  Index counts lookup
    Keys: (0.SAMPLE_AC = 'Xxxx') AND (0.SAMPLE_BX = 7001) AND (
          0.SAMPLE_CD = '1234567') AND (0.SAMPLE_XC = 'Yyyy')
~Estim  SAMPLE_KEY Hashed: Nodes=0, Est=4 Precise IO=0
~Estim  SAMPLE_RANKED_1 Ranked: Nodes=1, Min=2, Est=2 Precise IO=1
~Estim  RLEAF Cardinality=  1.4922310E+06
~Estim  SAMPLE_RANKED_2 Ranked: Nodes=1, Min=2, Est=2 Precise IO=1
~E#0002.01(1) Estim   Index/Estimate 2/2 3/2 1/4
~E#0002.01(1) BgrNdx2 EofData  DBKeys=0  Fetches=0+0  RecsOut=0 #Bufs=0

           0
1 row selected
SQL>
```

For such a strategy to be used, multiple SORTED RANKED indices must include leading segments that match the query criteria specified by the WHERE clause.

This problem can be avoided by using SET FLAGS 'NOCOUNT_SCAN' prior to executing this query, either as a dynamic statement or by defining the logical name RDMS$SET_FLAGS.

This problem has been corrected in Oracle Rdb Release 7.3.1.2. Oracle Rdb now supports multiple SORTED

RANKED indices for use with *Index counts lookup* optimization as part of a dynamic query strategy. In addition, the STRATEGY display now includes added notation to show that *Index counts lookup* optimization was used.

2.2.2 Unexpected −RDMS−F−ACTQUERY Error During ALTER TABLE

Bug 13738014

In prior releases of Oracle Rdb, an ALTER TABLE statement might fail if a DEFAULT clause referenced a SQL function that also referenced the table being altered. The following example shows the reported error.

```
create table T
    (id integer
    );

create module M
    function MAX_ID ()
    returns integer;
    return (select max (id) from T) + 1;
end module;

alter table T
    add column ID2 integer default MAX_ID();
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−F−ACTQUERY, there are queries compiled that reference relation "T"
−RDMS−F−RELNOTCHG, relation T has not been changed
```

This problem may occur for ALTER TABLE ... ALTER COLUMN or ALTER TABLE ... ADD COLUMN clauses.

This problem has been corrected in Oracle Rdb Release 7.3.1.2. SQL now avoids loading the referenced function when checking the return result type of the referenced function. This also means that nested routines are no longer loaded.

2.2.3 Unexpected SQL Bugcheck With Malformed INSERT or UPDATE Column Target

Bug 18848877

In prior releases of Oracle Rdb, SQL would bugcheck if the target of an INSERT or UPDATE was erroneously coded as a sequence reference (either CURRVAL or NEXTVAL).

The following example shows the problem.

```
SQL> insert into X (y.nextval) values (0);
%RDMS−I−BUGCHKDMP, generating bugcheck dump file USER2:[TESTER]SQLBUGCHK.
%SQL−F−BUGCHK, There has been a fatal error. Please contact your Oracle support
representative
SQL> update X set y.nextval = 0;
%RDMS−I−BUGCHKDMP, generating bugcheck dump file USER2:[TESTER]SQLBUGCHK.
%SYSTEM−F−ACCVIO, access violation, reason mask=00, virtual address=000000000000
```

This problem has been corrected in Oracle Rdb Release 7.3.1.2. SQL now correctly diagnoses these problems. The following output from the SQL Module Language compiler shows the new diagnostics.

```
insert into X (y.currval) values (1);
               1
%SQL−F−INVCOLREF, (1) Invalid column reference for INSERT or UPDATE statement
update X set y.nextval = 0;
             1
%SQL−F−INVCOLREF, (1) Invalid column reference for INSERT or UPDATE statement
```

2.2.4 Incorrect Evaluation of DEFAULT Expression During INSERT Statement

Bug 18999785

In prior versions of Oracle Rdb, it was possible to define a DEFAULT which referenced another column in the same table but not have that DEFAULT correctly evaluated during the INSERT statement.

The following example shows that the DEFAULT inherited by column C3 is incorrect. It should be NULL.

```
SQL> create table t1
cont>     (c1 int
cont>     ,c2 char (2)
cont>     ,c3 char (2) default lower(c2)
cont>     );
SQL>
SQL> insert into t1 (c1) values (4);
1 row inserted
SQL>
SQL> select * from t1;
          C1   C2     C3
           4   NULL   ..
1 row selected
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.1.2.

2.3 RMU Errors Fixed

2.3.1 RMU Unload Not Generating Oracle Style INTERVAL DAY Values

Bug 18262460

In prior versions of Oracle Rdb, the RMU Unload command, when requested to generate a CONTROL file format for processing by the sqlldr tool, would not generate INTERVAL DAY values acceptable by the Oracle Database.

This problem has been corrected in Oracle Rdb Release 7.3.1.2. RMU Unload now implicitly establishes the DIALECT as ORACLE LEVEL2 (see SET DIALECT Command for more details). This dialect causes Oracle Rdb to use an ASCII space character to separate the DAY field from the following HOUR field.

In addition, RMU Unload also transforms the following types to be compatible with the Oracle Database.

- DATE VMS is unloaded as TIMESTAMP(2).
  This allows fractional second values (100th of a second) to be unloaded. In prior releases, this type was unloaded as a DATE with fractional second values truncated.
- INTERVAL YEAR and INTERVAL MONTH types are unloaded as INTERVAL YEAR(9) TO MONTH.
- INTERVAL DAY, INTERVAL HOUR, INTERVAL MINUTE, INTERVAL SECOND, INTERVAL DAY TO HOUR, INTERVAL DAY TO MINUTE, INTERVAL HOUR TO MINUTE, INTERVAL HOUR TO SECOND, INTERVAL MINUTE TO SECOND are unloaded as INTERVAL DAY(9) TO SECOND(2).

2.3.2 Full Backup No Longer Required After Altering the Snapshot Area Page Allocation

Bug 18319063

Modifying the Oracle Rdb database snapshot storage area page allocation previously required the next database backup to be a full database backup. If an incremental backup was executed without a preceding full backup following a modification of the snapshot storage area page allocation, the fatal RMU−F−NOFULLBCK error message was output and the incremental backup was aborted.

This restriction has now been removed since database corruption does not occur when an incremental database restore is not preceded by a full database restore following a modification of the database snapshot storage area page allocation. When the snapshot storage area page allocation is modified, the incremental backup now succeeds when the incremental backup is executed without a preceding full backup.

The following example shows the previous incorrect behavior. If the next database backup after altering a database storage area snapshot page allocation was not a full backup, the %RMU−F−NOFULLBCK message was output and the backup was aborted.

```
$ SQL
  alter database filename mf_personnel
  alter storage area jobs snapshot allocation is 100 pages;
  exit
$ rmu/backup/incremental/nolog mf_personnel.rdb -
  DEVICE:[DIRECTORY]mfp.rbf
%RMU-F-NOFULLBCK, no full backup of this database exists
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 28-FEB-2014 06:56:02.11
$
```

The following example shows that this restriction has now been removed. If the next database backup after altering a database storage area snapshot page allocation is an incremental backup, the incremental backup now succeeds.

```
$ SQL
  alter database filename mf_personnel
  alter storage area jobs snapshot allocation is 100 pages;
  exit
$ rmu/backup/incremental/nolog mf_personnel.rdb -
  DEVICE:[DIRECTORY]mfp.rbf
$
```

This problem has been corrected in Oracle Rdb Release 7.3.1.2.

2.3.3 RMU Unload After_Journal Did Not Correctly Unload Oracle Database Format Date/Time Values

Bug 18262460

In prior releases of Oracle Rdb, RMU/UNLOAD/AFTER_JOURNAL would generate incorrect definitions in the generated CONTROL file (.CTL) and incorrectly formatted data for date/time types (DATE, TIME, TIMESTAMP and INTERVAL).

This problem has been corrected in Oracle Rdb Release 7.3.1.2. RMU/UNLOAD/AFTER_JOURNAL now generates definitions and data that are acceptable to Oracle Database release 10g or later and allow these complex types to be shipped in a portable data format and used by SQL*Loader (sqlldr).

These changes include:

- DATE VMS is unloaded as TIMESTAMP(2). This allows fractional second values (100th of a second) to be unloaded. In prior releases, this type was unloaded as DATE with fractional second values truncated.
- INTERVAL YEAR and INTERVAL MONTH types are unloaded as INTERVAL YEAR(9) TO MONTH.
- INTERVAL DAY, INTERVAL HOUR, INTERVAL MINUTE, INTERVAL SECOND, INTERVAL DAY TO HOUR, INTERVAL DAY TO MINUTE, INTERVAL HOUR TO MINUTE, INTERVAL HOUR TO SECOND, INTERVAL MINUTE TO SECOND are unloaded as INTERVAL DAY(9) TO SECOND(2).

2.3.4 RMU Convert Leaves After Image Journal SUPPRESSED

Bug 19429891

In prior versions of Oracle Rdb V7.3, it was possible that RMU Convert would leave a journal in the SUPPRESSED state, without warning from RMU Convert.

```
$ rmu/convert/noconfirm sample
%RMU−I−RMUTXT_000, Executing RMU for Oracle Rdb V7.3−110 on OpenVMS IA64 V8.4
%RMU−I−AIJ_DISABLED, after−image journaling is being disabled temporarily for
the Convert operation
%RMU−I−LOGCONVRT, database root converted to current structure level
%RMU−S−CVTDBSUC, database USER2:[TESTING]SAMPLE.RDB;1 successfully converted
from version V7.2 to V7.3
%RMU−I−CVTCOMSUC, CONVERT committed for USER2:[TESTING]SAMPLE.RDB;1 to version
V7.3
%RMU−I−LOGMODSTR,     activated after−image journal "SAMPLE_JOURNAL_01"
%RMU−W−DOFULLBCK, full database backup should be done to ensure future recovery
```

The /NOCONFIRM was used to avoid the questions from RMU Convert concerning complete backups prior to the convert of the database.

This change to the journal state would later be diagnosed when an RMU Backup After_Image was perform. The following example shows that the error RMU−F−AIJBCKINAC is reported during the backup of the after image journals.

```
$ rmu/backup/after sample ""
%RMU−I−AIJBCKBEG, beginning after−image journal backup operation
%RMU−I−OPERNOTIFY, system operator notification: AIJ backup operation started
%RMU−I−AIJBCKSEQ, backing up after−image journal sequence number 1
%RMU−I−LOGBCKAIJ, backing up after−image journal SAMPLE_JOURNAL_02 at
17:22:08.86
%RMU−I−AIJBCKSTOP, backup of after−image journal SAMPLE_JOURNAL_02 did not
complete
%RMU−I−OPERNOTIFY, system operator notification: AIJ manual backup operation
failed
%RMU−F−AIJBCKINAC, AIJ backup completed when accessing inaccessible journal SAMPLE_JOURNAL_02
%RMU−F−FTL_BCK, Fatal error for BACKUP operation at  4−SEP−2014 17:22:08.86
```

This problem has been corrected in Oracle Rdb Release 7.3.1.2. Both RMU Convert and RMU Backup have been modified to issue improved diagnostics in this case. In addition the *Oracle Rdb Installation and Configuration Guide* has been updated with instructions for recovering from this state.

In this example, an RMU Set After_Image statement can be used to modify the named after image to the current Oracle Rdb version. Oracle recommends that the contents of the after image file be backed up prior to the RMU/SET AFTER_IMAGE command since it will modify and truncate the prior contents.

```
$ rmu/set after/alter=(Name=SAMPLE_JOURNAL_02) SAMPLE
%RMU−I−LOGMODSTR,     unsuppressed after−image journal "SAMPLE_JOURNAL_02"
```

The following example shows the RMU−W−AIJSUPPRESSED now issued by RMU Convert for each suppressed journal.

```
$ rmu/convert/noconfirm sample
%RMU−I−RMUTXT_000, Executing RMU for Oracle Rdb V7.3−120 on OpenVMS IA64 V8.4
%RMU−I−AIJ_DISABLED, after−image journaling is being disabled temporarily for
the Convert operation
%RMU−I−LOGCONVRT, database root converted to current structure level
%RMU−S−CVTDBSUC, database USER2:[TESTING]SAMPLE.RDB;1 successfully converted
from version V7.2 to V7.3
%RMU−I−CVTCOMSUC, CONVERT committed for USER2:[TESTING]SAMPLE.RDB;1 to version
V7.3
%RMU−I−LOGMODSTR,     activated after−image journal "SAMPLE_JOURNAL_01"
%RMU−W−AIJSUPPRESSED, After journal "SAMPLE_JOURNAL_02" has been
suppressed and not converted since it contains data from the previous
version
%RMU−W−DOFULLBCK, full database backup should be done to ensure future recovery
```

In addition, RMU Backup has been modified to similarly warn that some of the after image journal files (AIJ) are incompatible with the current version. The subsequent database backup is unaffected by the state of the after image journals.

```
$ rmu/backup sample sample_bck73
%RMU−F−BADAIJVER, after−image journal version is incompatible with
the runtime system
%RMU−W−ROOERRORS,        1 error encountered in root verification
%RMU−I−BCKTXT_00, Backed up root file USER2:[TESTING]SAMPLE.RDB;3
%RMU−I−BCKTXT_02, Starting full backup of storage area (RDB$SYSTEM)
USER2:[TESTING]SAMPLE.RDA;3 at  4−SEP−2014 17:22:08.80
%RMU−I−BCKTXT_12, Completed full backup of storage area (RDB$SYSTEM)
USER2:[TESTING]SAMPLE.RDA;3 at  4−SEP−2014 17:22:08.81
%RMU−I−BCKTXT_02, Starting full backup of storage area (SAMPLE_AREA1)
USER2:[TESTING]SAMPLE_AREA1.RDA;3 at  4−SEP−2014 17:22:08.81
%RMU−I−BCKTXT_12, Completed full backup of storage area (SAMPLE_AREA1)
USER2:[TESTING]SAMPLE_AREA1.RDA;3 at  4−SEP−2014 17:22:08.82
%RMU−I−COMPLETED, BACKUP operation completed at  4−SEP−2014 17:22:08.82
```

2.4 RMU Show Statistics Errors Fixed

2.4.1 RMU/SHOW STATISTICS Playback Zeroed Final Transaction Duration Screen

Bug 14549459

On the OpenVMS Itanium platform, if an Oracle Rdb RMU/SHOW STATISTICS/INPUT command was invoked to play back a prerecorded binary statistics file created by a prior RMU/SHOW STATISTICS/OUTPUT command, and a TRANSACTION DURATION (TOTAL) screen was invoked, when the playback ended the TRANSACTION DURATION (TOTAL) screen statistics were all incorrectly set to zeroes. This did not happen on the OpenVMS Alpha platform.

This problem has been fixed. Now, after the RMU/SHOW STATISTICS playback reaches the end of file, the TRANSACTION DURATION (TOTAL) screen statistics displayed will have valid values.

The following example shows the problem. At the end of the playback, the TRANSACTION DURATION (TOTAL) screen statistics were all zero.

```
$  rmu/show statistics/time=-1/input=RMU_STAT.DAT

Node: A (1/1/2)                 Oracle Rdb V7.3-100 Perf. Monitor
                                     7-AUG-2013 23:59:25.65
Rate: 0.01 Seconds              Transaction Duration (Total)
                                     Elapsed: 2 16:59:08.19
Page: 1 of 1
DEV:[DIR]TEST.RDB;1                                    Mode: Replay

Total transaction count:            0
Seconds    Tx.Count:   %  #Complete:   % #Incomplete:   %
 0-< 1:           0   0%          0   0%           0   0% <-avg=0.000000 95%=0.00
 1-< 2:           0   0%          0   0%           0   0%
 2-< 3:           0   0%          0   0%           0   0%
 3-< 4:           0   0%          0   0%           0   0%
 4-< 5:           0   0%          0   0%           0   0%
 5-< 6:           0   0%          0   0%           0   0%
 6-< 7:           0   0%          0   0%           0   0%
 7-< 8:           0   0%          0   0%           0   0%
 8-< 9:           0   0%          0   0%           0   0%
 9-<10:           0   0%          0   0%           0   0%
 10+++:           0   0%          0   0%           0   0%
```

The following example shows that the problem has been fixed. At the end of the playback, the TRANSACTION DURATION (TOTAL) screen statistics contain valid values.

```
$  rmu/show statistics/time=-1/input=RMU_STAT.DAT

Node: A (1/1/2)                   Oracle Rdb V7.3-120 Perf. Monitor
                                         7-JAN-2014 23:59:25.65
Rate: 0.01 Seconds                Transaction Duration (Total)
                                         Elapsed: 2 16:59:08.19
Page: 1 of 1
DEV:[DIR]TEST.RDB;1                                    Mode: Replay
```

```
Total transaction count:    165498168
Seconds    Tx.Count:   %  #Complete:   % #Incomplete:   %
 0−< 1:   165399645  99%  165399645  99%       98523   1% <−avg=0.011572 95%=0.01
 1−< 2:       55503   0%  165455148  99%       43020   1%
 2−< 3:       15124   0%  165470272  99%       27896   1%
 3−< 4:        6768   0%  165477040  99%       21128   1%
 4−< 5:        3512   0%  165480552  99%       17616   1%
 5−< 6:        2870   0%  165483422  99%       14746   1%
 6−< 7:        1764   0%  165485186  99%       12982   1%
 7−< 8:        1729   0%  165486915  99%       11253   1%
 8−< 9:        1562   0%  165488477  99%        9691   1%
 9−<10:        1106   0%  165489583  99%        8585   1%
 10+++:        8585   0%  165498168 100%           0   0%
```

This problem was actually corrected in Oracle Rdb Release 7.3.1.1.

2.4.2 Field Help Missing for Some RMU Show Statistics Fields

Bug 19267444

In prior releases of Oracle Rdb, it was possible that some field level help for some RMU Show Statistics screens was missing. This has been corrected for the reported cases.

This problem has been corrected in Oracle Rdb Release 7.3.1.2. If any missing field help is noticed, Oracle requests that you contact Oracle Support after running the RMU/SHOW STATISTICS command with /OPTION=DEBUG and repeating the Help request. RMU will display the screen and field keys with the "No help available" message. This information will be used by Oracle Rdb engineering to locate and correct the missing help text.

Chapter 3
Software Errors Fixed in Oracle Rdb Release 7.3.1.1

This chapter describes software errors that are fixed by Oracle Rdb Release 7.3.1.1.

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 Memory Leak Corrected in Queries That Use Sorting

Bug 17360970

In all prior releases of Oracle Rdb, there was a minor memory leak that occurred when a query used sorting (ORDER BY, GROUP BY, DISTINCT, UNION, and so on) and the query was terminated prior to returning all rows. For instance, a cursor was closed (using CLOSE, COMMIT or RELEASE statements) or if the query contained a LIMIT TO clause.

In these cases, the normal cleanup code could fail to release a small piece of virtual memory if an I/O had fetched data from the SORT work files and that buffered data was not returned to the query.

Workarounds to this problem include:

- Avoiding the use of SORT for such queries. For example, creating an index (possibly adding a COLLATING SEQUENCE) which is the correct order for the query. In such cases, the SORT operation can be avoided by using the index ordering.
- Avoid using the disk sorting interface. This can be done by allowing QSORT (an alternate algorithm) to be used for larger sets of data or defining the logical name RDMS$BIND_MAX_QSORT_COUNT to a value adequate to hold all rows input into the sort.
- Restructuring the query so that the LIMIT TO is part of an outer reference and perform the ORDER BY in a nested derived table.

This problem has been corrected in Oracle Rdb Release 7.3.1.1.

3.1.2 Wrong Result Using Match Strategy on EXISTS Subquery

Bug 16819636

In prior releases of Oracle Rdb, the optimizer might return wrong results when the previous aggregate subquery is later joined by a cross with a direct lookup index key, which is part of an equivalent class of transitive equality booleans between multiple table joins.

For example, the following query contains four tables joined with transitive equality booleans between them, followed by an EXISTS subquery joining with another equality boolean transitive to one of the outer tables.

```
SELECT
    t2.active, t2.fdate, t1.id, t1.status
FROM
    T0 t0, T1 t1, T2 t2, T3 t3,
WHERE
    t1.status = 'E'
    AND ( (t3.fdate BETWEEN '20130421' AND '20130810') OR (t3.active='1') )
    AND t3.id=t1.id
    AND EXISTS
      (SELECT * FROM T4 t4 WHERE
          t4.id=t1.id AND t4.type = '1')
    AND t2.id=t1.id
    AND t2.id=t0.id
    AND ((t2.fdate between '20130421' AND '20130810') OR (t2.active='1'))
    AND t0.iso='GB'
;
Tables:
  0 = T0
  1 = T1
  2 = T2
  3 = T3
  4 = T4
Cross block of 4 entries  Q1
  Cross block entry 1
    Conjunct: 3.ID = 1.ID
    Match  Q1
      Outer loop
        Match_Key:1.ID
        Conjunct: <agg0> <> 0
        Match  Q1
          Outer loop      (zig-zag)
              Match_Key:1.ID
              Index_Key:ID
                Conjunct: 1.STATUS = 'E'
                Get     Retrieval by index of relation 1:T1
                  Index name  T1_IDX [0:0]
              Inner loop      (zig-zag)
              Match_Key:4.ID
              Index_Key:TYPE, ID, FDATE
                Aggregate-F1: 0:COUNT-ANY (<subselect>) Q2
                Index only retrieval of relation 4:T4
                  Index name  T4_IDX [1:1]
                    Keys: T4.TYPE = '1'
      Inner loop      (zig-zag)
        Match_Key:3.ID
        Index_Key:ID, SLID, FDATE
          Conjunct: ((3.FDATE >= '20130421') AND (3.FDATE <=
                      '20130810')) OR (3.ACTIVE = '1')
          Get     Retrieval by index of relation 3:T3
            Index name  T3_IDX [0:0]
  Cross block entry 2
    Index only retrieval of relation 0:T0
      Index name  T0_NDX [2:2]        Direct lookup <== See Note below
        Keys: (4.ID = 0.ID) AND (0.ISO = 'GB')
  Cross block entry 4
    Leaf#01 FFirst 2:T2 Card=32
      Bool: (2.ID = 1.ID) AND (2.ID = 0.ID) AND (((2.FDATE >= '20130421'
            ) AND (2.FDATE <= '20130810')) OR (2.ACTIVE = '1'))
      BgrNdx1 T2_IDX [1:1] Fan=11
        Keys: 2.ID = 1.ID
...etc...

0 rows selected   <= wrong result returned
```

The wrong result is caused by the index boolean "(4.ID = 0.ID)" in the keys for the direct lookup index T0_NDX under the "Cross block entry 2".

A workaround may be to use a query outline to change the join order.

These problems have been corrected in Oracle Rdb Release 7.3.1.1.

3.1.3 Bugcheck at DIOFETCH$FETCH_SNAP_SEG

Bug 8881798

Starting in Oracle Rdb Release V7.2, it was possible for a read−only transaction to bugcheck with an exception at routine DIOFETCH$FETCH_SNAP_SEG. The actual offset within that routine would depend on the current platform and release version number.

For database consistency, if a record is modified on a data page by a read−write (RW) transaction, but not committed prior to the start of a read−only (RO) transaction, those updates should not be visible to that RO transaction. In such a case, the RO transaction would need to find some prior version of the record that would be visible. This prior version is written by a RW transaction to a page in a snapshot file. The data page maintains a pointer to this snapshot page. Transaction Sequence Numbers (TSNs) play an important role in determining which prior copy of a record is visible and when a snapshot page is considered obsolete and re−useable.

The bugcheck occurs when the RO transaction needs to find a prior version of a data record, follows the pointer to one or more snapshot pages, but cannot find a version that is visible. This problem was caused by RW transactions that erroneously re−used valid snapshot pages needed by these RO transactions.

This problem does not cause any data inconsistencies or corruption. Simply restarting the RO transaction would often suffice.

This problem has been corrected in Oracle Rdb Release 7.3.1.1.

3.1.4 Performance Improvement for OJ Query With Temporary TTBL and Sort

Bug 3539303

In prior releases of Oracle Rdb, an Outer Join query might slow down when the inner loop applies a temporary table with sort node. See the following examples.

1. Left OJ query with "is null" predicate

```
select count(*)
    from T1 as C1
        left outer join
            T2 as C2
                 on (C1.COL_ID = C2.COL_ID)
    where C2.COL_ID is null;
Tables:
  0 = T1
  1 = T2
Aggregate: 0:COUNT (*) Q2
Conjunct: MISSING (1.COL_ID)
Match    (Left Outer Join)  Inner_TTBL Q3
  Outer loop
  Match_Key:0.COL_ID
    Index only retrieval of relation 0:T1
```

```
      Index name  I_AR_AT_2 [0:0]
  Inner loop
  Match_Key:1.COL_ID
    Temporary relation
    Sort: 1.COL_ID(a)
    Index only retrieval of relation 1:T2
      Index name  I_UR_TN_AS_6 [0:0]

           0
1 row selected
```

2. Left OJ query with "CCOL = CVAR" predicate

```
select count(*)
    from T1 as C1
        left outer join
            T2 as C2
                 on (C1.COL_ID = C2.COL_ID)
    where C2.COL_ID = 0;
Tables:
  0 = T1
  1 = T2
Aggregate: 0:COUNT (*) Q2
Conjunct: 1.COL_ID = 0
Match    (Left Outer Join)  Inner_TTBL Q3
  Outer loop
  Match_Key:0.COL_ID
    Index only retrieval of relation 0:T1
      Index name  I_AR_AT_2 [0:0]
  Inner loop
  Match_Key:1.COL_ID
    Temporary relation
    Sort: 1.COL_ID(a)
    Conjunct: 1.COL_ID = 0
    Index only retrieval of relation 1:T2
      Index name  I_UR_TN_AS_6 [1:1]
        Keys: 1.COL_ID = 0

           0
1 row selected
```

A workaround may be to use a query outline to change the match strategy to a cross strategy.

This problem has been corrected in Oracle Rdb Release 7.3.1.1.

3.1.5 In Some Cases System User Audited Instead of Session User

Bug 17731257

In prior releases of Oracle Rdb, it was possible that when auditing actions on database objects, the SYSTEM_USER was recorded instead of the SESSION_USER. This occurred during user impersonation, such as when an application uses CONNECT using USER and USING parameters.

This most commonly occurs when the SESSION_USER is not established for auditing database events. A possible workaround is to enable DACCESS auditing for the DATABASE for at least the SELECT privilege.

```
$ RMU/SET AUDIT/ENABLE=DACCESS=DATABASE/PRIV=(SELECT,ALTER) PERSONNEL
```

This problem has been corrected in Oracle Rdb Release 7.3.1.1. Oracle Rdb now correctly records the session user in the audit journal.

3.1.6 Using BEGIN/END Around Set Transaction Leads to Memory Leak

Bug 16536178

The following sequence of SQL statements can cause a memory leak in Dispatch and, if run for long enough without disconnecting, could cause the process to run out of memory.

```
BEGIN
  SET TRANSACTION READ WRITE
    RESERVING EMPLOYEES FOR SHARED WRITE;
END;
BEGIN
  COMMIT;
END;
```

Removing at least one BEGIN/END (it doesn't matter which) can avoid the problem.

This problem has been corrected in Oracle Rdb Release 7.3.1.1.

3.1.7 Excessive Alignment Faults on Client Side Using RDB$REMOTE

Bug 18155052

It was possible for the client side of an RDB$REMOTE type of connection to experience excessive alignment faults.

```
Exception PC       Rate  Exception PC         Module      Offset    EPID
-----------------  ----  --------------------------------------------------
00000000.800F5191  94.95 RDB$SHARE72+8005D191 RDB$SHARE72 8005D191   20E07125
00000000.800F50B0  94.86 RDB$SHARE72+8005D0B0 RDB$SHARE72 8005D0B0   20E07125
```

This problem has been corrected in Oracle Rdb Release 7.3.1.1.

3.1.8 Ranked Index Bugchecks − PSII2REMOVEDUPBBC & PSII2INSERTDUPBBC

Bug 17383599

In prior versions of Oracle Rdb, applications trying to insert or remove records using a ranked index may bugcheck with an exception similar to the following:

```
Exception at 00000000xxxxxxxx : RDMSHRP72\PSII2REMOVEDUPBBC + 0000xxxx
%COSI−F−BUGCHECK, internal consistency failure
```

A bugcheck may also be raised with the following exception:

```
Exception at 00000000xxxxxxxx : RDMSHRP72\PSII2INSERTDUPBBC + 0000xxxx
%COSI−F−BUGCHECK, internal consistency failure
```

In addition, the following exception may be raised during record retrieval:

```
%RDMS−F−NODBK, 61:117:29 does not point to a data record
```

This problem may also lead to a corruption of the index on−disk: an RMU VERIFY INDEX of the database will indicate that the index is corrupt with an exception similar to:

```
RMU−E−BADDBKFET, Error fetching dbkey 61:117:29
```

Depending on the transaction mix of insertions and deletions, it is possible that these problems may not cause any on−disk index corruption: the problem may only be a transient error, affecting in−memory structures only.

These problems are more likely to occur when there are a large number of duplicates within the ranked index entries.

A possible workaround is to rebuild the index if RMU/VERIFY has shown an exception similar to the one cited above. The rebuild of the index will rectify any ranked index corruptions. However, in prior versions of Oracle Rdb, the same problem may re−occur on subsequent transactions.

These problems have been corrected in Oracle Rdb Release 7.3.1.1.

3.2 SQL Errors Fixed

3.2.1 Memory Leak Possible in DESCRIBE Dynamic SQL Statement

Bug 14570971

In prior releases of Oracle Rdb V7.3, it was possible that applications using Dynamic SQL could encounter a small memory leak. Application servers which stayed active for a long time might exhaust available memory. This might affect SQL/Services, JDBC, and so on.

This problem occurs during the DESCRIBE statement when processing any of several special character columns or functions. This includes:

- DBKEY
- ROWID
- the SYS_GUID() function
- the RDB$SECURITY_CLASS system table column
- a column defined using the system domain RDB$DATABASE_KEY
- any COMPUTED BY or AUTOMATIC AS column that uses such value expressions
- and any view column based on any of these value expressions.

These character types use a special character set which is not being handled correctly when building the SQLDA.

This problem has been corrected in Oracle Rdb Release 7.3.1.1.

3.2.2 Changes to Date/Time Literal Processing

- This release of Oracle Rdb improves the diagnostics reported for malformed date/time and interval values. In prior versions of Oracle Rdb, a single exception COSI$_IVTIME would be reported if there was a syntax error in the date/time value or if the field values exceeded the allowed range of values. The following example shows these additional messages.

  ```
  SQL> select interval'100−100' year to month from rdb$database;
  %SQL−F−DATCONERR, Data conversion error for string '100−100'
  −COSI−F−IVTIMEILF, invalid interval − leading field has too many digits
  SQL> select interval'100−100' year(3) to month from rdb$database;
  %SQL−F−DATCONERR, Data conversion error for string '100−100'
  −COSI−F−IVTIMEINT, invalid interval − error in the format or values
  SQL> select time'12:00:01.999' from rdb$database;
  %SQL−F−DATCONERR, Data conversion error for string '12:00:01.999'
  −COSI−F−IVTIMEFSP, invalid time or interval − fractional seconds field has too
  many digits
  SQL>
  ```

  Oracle Rdb now returns SQLSTATE '22015' *Interval field overflow* in some cases where SQLSTATE '22007' *Invalid datetime format* was returned in prior releases.

- Leading zeros for the interval leading field are now ignored when checking the format of literal values. Prior versions of Oracle Rdb would raise an exception even though these were insignificant.

  ```
  SQL> create table TEST (duration interval year(2) to month);
  SQL> insert into TEST values (interval'00000'year to month);
  %SQL−F−DATCONERR, Data conversion error for string '00000'
  −COSI−F−IVTIMEILF, invalid interval − leading field has too many digits
  −COSI−F−IVTIME, invalid date or time
  ```

  This example no longer causes an exception.
- This release of Oracle Rdb also allows the "T" character as a separator between date and time portions of a TIMESTAMP literal. This allows literal dates that conform to *ISO 8601 Data elements and interchange formats − Information interchange − Representation of dates and times* which is an International standard for date/time representations.

  ```
  SQL> select timestamp'2013−10−29T14:52:11.42' from rdb$database;

   2013−10−29 14:52:11.42
  1 row selected
  ```
- New SQLCODE values
  The table *Values Returned to the SQLCODE Field* now includes these new SQLCODE values.

  Table 3−1 Values Returned to the SQLCODE Field

Numeric Value	Literal Value	Meaning
−1045	SQLCODE_INV_INTERVAL	Invalid interval format
−1046	SQLCODE_INV_FRACSEC	Time, Timestamp or interval has too many fractional digits
−1047	SQLCODE_INV_INTLEAD	Interval leading field is too large
−1048	SQLCODE_INC_CSET	Incompatible character set
−1049	SQLCODE_DATA_CVT_ERROR	Data conversion error

These problems have been corrected in Oracle Rdb Release 7.3.1.1.

3.2.3 Unexpected Bugcheck From Invalid DBKEY Use

Bug 17656896

In prior releases of Oracle Rdb, the test for valid DBKEY (ROWID) values might fail to diagnose some illegal values. The result was a bugcheck when an attempt was made to use these bad DBKEY values.

The following example shows one such case.

```
SQL> begin
cont> set :dbk = _dbkey'−32767:1:0';
cont> end;
SQL>
SQL> select * from employees
cont>  where dbkey = :dbk;
```

```
%RDMS−I−BUGCHKDMP, generating bugcheck dump file USER2:[...]RDSBUGCHK.DMP;
%SYSTEM−F−ACCVIO, access violation, reason mask=00,
virtual address=0000000000000000, PC=FFFFFFFF8191FA30, PS=0000001B
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.1.1. The DBKEY validation has been improved to catch this and similar cases. This is shown in the following example.

```
SQL> select * from employees
cont>  where dbkey = :dbk;
%RDB−E−NO_RECORD, access by dbkey failed because dbkey is no longer associated
with a record
−RDMS−F−NODBK, −32767:1:0 does not point to a data record
SQL>
```

3.2.4 Not Equals Operator Causes BITMAPPED SCAN Strategy to be Rejected

Bug 5399245

In some cases, the use of *not equals* will cause the Rdb optimizer to reject BITMAPPED SCAN strategy. This is shown in this example:

```
SQL> select count(*)
cont> from
cont>   (select dbkey from T1 where C2 <>'XYZ' or C4 >='03−jul−2006' ) as x
cont> optimize for bitmapped scan;
Tables:
  0 = T1
Aggregate: 0:COUNT (*) Q2
Merge of 1 entries  Q2
  Merge block entry 1  Q3
  Conjunct: (0.C2 <> 'XYZ') OR (0.C4 >= '3−JUL−2006')
  Get     Retrieval sequentially of relation 0:T1

                 2514
1 row selected
SQL>
```

With this release of Oracle Rdb, when BITMAPPED SCAN is requested and REWRITE is enabled, the query compiler will rewrite the *not equals* operator (<>) as an OR of a *less than* operator (<) and a *greater than* operator (>). This new expression is semantically the same and encourages the use of the BITMAPPED SCAN strategy. Note that this transformation is only used if one of the operands of the *not equals* operator is a column of a sorted index.

Programmers can request bitmapped scan be attempted using the SET FLAGS statement, the RDMS$SET_FLAGS logical name, the RDMS$ENABLE_BITMAPPED_SCAN logical name or the OPTIMIZE clause of the query.

This is the new strategy using the new capability of Oracle Rdb. The result is a faster query using less I/O and lower CPU time.

```
SQL> select count(*)
cont> from
cont>   (select dbkey from T1 where C2 <>'XYZ' or C4 >='03-jul-2006' ) as x
cont> optimize for bitmapped scan;
Tables:
  0 = T1
Aggregate: 0:COUNT (*) Q2
Merge of 1 entries  Q2
  Merge block entry 1  Q3
  Leaf#01 BgrOnly 0:T1 Card=348838         Bitmapped scan
    Bool: (0.C2 < 'XYZ') OR (0.C2 > 'XYZ') OR (0.C4 >= '3-JUL-2006')
    BgrNdx1 X1_T1 [0:1,1:0] Fan=100
      Keys: r0: 0.C2 > 'XYZ'
            r1: 0.C2 < 'XYZ'
        OrNdx1 X3_T1 [1:0] Fan=84
          Keys: 0.C4 >= '3-JUL-2006'

                 2514
1 row selected
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.1.1.

3.2.5 Data Dictionary Tables Now Use Key Suffix Compression

Bug 18118478

In prior versions of Oracle Rdb, when preparing a database for use by OCI Services for Rdb, indices for the data dictionary tables were created with "Key suffix compression is DISABLED".

The following example shows this setting for the index on the USER$ table.

```
$ @sys$share:RDB_NATCONN73.COM
Operation (prepare/upgrade/drop/add_user/modify_user/remove_user/show_users):
prepare
Database: mf_personnel
Developer version (60/6I/<CR>):
**** Preparing database MF_PERSONNEL ****
**** Preparing database successfully completed ****
Operation (prepare/upgrade/drop/add_user/modify_user/remove_user/show_users):
$
$ sql$
SQL> attach 'file mf_personnel';
SQL> show table USER$
...
Indexes on table USER$:
USER$_NAME                      with column NAME
 No Duplicates allowed
 Type is Sorted
 Key suffix compression is DISABLED
 Node size 430
...
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.1.1. Now such indices will be created with "Key suffix compression is ENABLED". This has the advantage of reducing the size of metadata indices, since most names are space filled to 31 octets.

3.2.6 Unexpected Bugcheck When Using NUMBER OF SWEEP ROWS Clause

Bug 18187722

In prior releases of Oracle Rdb, the clause NUMBER OF SWEEP ROWS was accepted as part of the ROW CACHE IS ENABLED clause. This would cause SQL to generate a bugcheck dump.

The following example shows the problem in ALTER DATABASE.

```
SQL> CREATE DATABASE
cont>     FILENAME 'SAMPLE.RDB'
cont>     ROW CACHE ENABLE
cont> ;
SQL> ALTER DATABASE
cont>     FILENAME 'SAMPLE.RDB'
cont>     ROW CACHE ENABLE
cont>           (NUMBER OF SWEEP ROWS IS 500);
%RDMS−I−BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]SQLBUGCHK.DMP;
%SQL−F−BUGCHK, There has been a fatal error. Please contact your Oracle
support representative. walk_alter_dbase − bad RC option
SQL>
```

A similar error occurs during CREATE DATABASE.

```
SQL> CREATE DATABASE
cont>     FILENAME 'SAMPLE.RDB'
cont>     ROW CACHE ENABLE
cont>           (NUMBER OF SWEEP ROWS IS 500
cont>           ,SWEEP INTERVAL IS 60 SECONDS);
%RDMS−I−BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]SQLBUGCHK.DMP;
%SQL−F−BUGCHK, There has been a fatal error. Please contact your Oracle
support representative. walk_create_dbase − bad RC option
```

This clause is only applicable to the CREATE CACHE, ALTER CACHE or ADD CACHE clauses and should not have been accepted or documented for the CREATE DATABASE statement, ALTER DATABASE statement or IMPORT Statement.

This problem has been corrected in Oracle Rdb Release 7.3.1.1.

3.2.7 Memory Leak Possible When Using BITMAPPED SCAN Queries

Bug 17834421

In prior releases of Oracle Rdb, applications which performed BITMAPPED SCAN on SORTED RANKED indices might encounter a small memory leak. This allocated but unused memory can accumulate in server processes that ATTACH (CONNECT) and DISCONNECT to many databases over time.

This problem is more obvious in Oracle Rdb Release 7.3.1 because all system tables use SORTED RANKED indices and are queried using BITMAPPED SCAN.

This problem has been corrected in Oracle Rdb Release 7.3.1.1. Memory used by BITMAPPED SCAN is now correctly released upon end of the query.

3.3 RMU Errors Fixed

3.3.1 RMU/CONVERT/ROLLBACK From V7.3 May Prevent Access to Some Tables

Bug 18020072

In some rare cases, an RMU/CONVERT/NOCOMMIT can damage the database metadata such that a subsequent RMU/CONVERT/ROLLBACK prevents access to some tables. An RMU/CONVERT/COMMIT however is not affected by this problem and the resulting Rdb V7.3 database is correctly converted.

The following example shows the reported error.

```
$ RMU/CONVERT/NOCOMMIT TEST
%RMU−I−RMUTXT_000, Executing RMU for Oracle Rdb V7.3−100 on OpenVMS IA64
V8.3−1H1
Are you satisfied with your backup of USER1:[TESTING]TEST.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 USER1:[TESTING]TEST.RDB;1 successfully converted from
 version V7.2 to V7.3
$ RMU/CONVERT/ROLLBACK TEST
%RMU−I−RMUTXT_000, Executing RMU for Oracle Rdb V7.3−100 on OpenVMS IA64
V8.3−1H1
Are you satisfied with your backup of USER1:[TESTING]TEST.RDB;1 and your backup
 of any associated .aij files [N]? y
%RMU−I−LOGCONVRT, database root converted to current structure level
%RMU−I−CVTROLSUC, CONVERT rolled−back for USER1:[TESTING]TEST.RDB;1 to version
V7.2
$ SQL$
SQL> attach 'file test';
SQL> show index T1_NDX
Indexes on table T1
T1_NDX                          with column COL1
%RDB−E−NO_RECORD, access by dbkey failed because dbkey is no longer associated
with a record
−RDMS−F−NODBK, 1:2847:130 does not point to a data record
```

This problem occurs while loading the index metadata for the referenced table. In practice, any query against the affected tables will fail. This includes tools such as RMU/UNLOAD, RMU/EXTRACT and SQL EXPORT DATABASE.

This problem has been corrected in Oracle Rdb Release 7.3.1.1. RMU/CONVERT now correctly manages the index attributes during conversion so that a rollback is no longer affected by changes to V7.3.

3.3.2 Bugcheck from RMU/VERIFY/ALL After Constraint Verification

Bug 17548158

In prior releases of Oracle Rdb, the RMU Verify command might fail with a bugcheck on a CHECK constraint similar to the example below.

```
check ((not col1 in (' ', NULL)))
```

If the IN clause does not contain NULL, then no bugcheck occurs.

In fact, the coding for this constraint is in error and does not achieve the expected behavior. A corrected version is shown here.

```
check ((col1 <> ' ' and col1 is not NULL))
```

This problem has been corrected in Oracle Rdb Release 7.3.1.1.

3.3.3 Unexpected Bugcheck During Large RMU Load When Using /Defer_Index_Updates Qualifier

Bug 13402630

In prior releases of Oracle Rdb, it was possible that RMU Load might fail with a bugcheck dump.

```
$ RMU/LOAD -
    ABC_DB -
    LARGE_TABLE -
    LARGE_TABLE_DATA.UNL -
    /Commit=100000 -
    /Defer_Index_Updates -
    /Transaction=exclusive -
    /Buffer=10000
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]RDSBUGCHK.DMP;
```

The workaround was to not use the /Defer_Index_Updates qualifier.

This problem has been corrected in Oracle Rdb Release 7.3.1.1.

3.3.4 RMU/BACKUP/AFTER_JOURNAL Returned a Success Status if %RMU−F−AIJJRNBSY

Bug 17316940

When the Oracle Rdb RMU/BACKUP/AFTER_JOURNAL command is backing up Rdb database After Image Journal files and the backup cannnot continue because RMU cannot access an AIJ file, it outputs the fatal error:

```
%RMU−F−AIJJRNBSY, journal AIJ_FILE_NAME is busy and cannot be backed up
```

The backup is then terminated. However, even though the RMU/BACKUP/AFTER_JOURNAL command was terminated because of a fatal error and did not complete, the backup operation returned a success exit status and the following output success message so that the %RMU−F−AIJJRNBSY failure status could be overlooked.

```
%RMU−I−AIJBCKEND, after−image journal backup operation completed successfully
```

This problem has been fixed. Now, when the fatal %RMU−F−AIJJRNBSY error condition occurs, the misleading message will no longer be output and the %RMU−F−AIJJRNBSY fatal error status will be set as the RMU/BACKUP/AFTER_JOURNAL command exit status instead of the incorrect success status.

The following example shows the problem. The after image journal file backup operation is terminated and the %RMU−F−AIJJRNBSY fatal error is output but the backup operation returns a success status and the following misleading message is output:

```
%RMU−I−AIJBCKEND, after−image journal backup operation completed successfully
```

Here is the example.

```
$ rmu/back/after_after/log mf_personnel mfpaijbck
%RMU−I−AIJBCKBEG, beginning after−image journal backup operation
%RMU−I−OPERNOTIFY, system operator notification: AIJ backup operation started
%RMU−I−AIJBCKSEQ, backing up after−image journal sequence number 0
%RMU−I−LOGBCKAIJ, backing up after−image journal AIJ1 at 10:04:53.86
%RMU−I−QUIETPT, waiting for database quiet point at 21−OCT−2013 10:04:53.86
%RMU−I−QUIETPTREL, released database quiet point at 21−OCT−2013 10:04:53.91
%RMU−F−AIJJRNBSY, journal AIJ1 is busy and cannot be backed up
%RMU−I−AIJNOBACKUP, AIJ contains no transactions that qualify for backup
%RMU−I−OPERNOTIFY, system operator notification: AIJ backup operation completed
%RMU−I−AIJBCKEND, after−image journal backup operation completed successfully
MALIBU>SHOW SYMBOL $STATUS
  $STATUS == "%X10000001"
```

The following example shows that this problem has been fixed. The fatal %RMU−F−AIJJRNBSY error is now the return status of the after image journal file backup operation and the misleading message is not output.

```
$ rmu/back/after/log mf_personnel mfpaijbck
%RMU−I−AIJBCKBEG, beginning after−image journal backup operation
%RMU−I−OPERNOTIFY, system operator notification: AIJ backup operation started
%RMU−I−AIJBCKSEQ, backing up after−image journal sequence number 0
%RMU−I−LOGBCKAIJ, backing up after−image journal AIJ1 at 10:04:53.86
%RMU−I−QUIETPT, waiting for database quiet point at 21−OCT−2013 10:04:53.86
%RMU−I−QUIETPTREL, released database quiet point at 21−OCT−2013 10:04:53.91
%RMU−F−AIJJRNBSY, journal AIJ1 is busy and cannot be backed up
%RMU−I−AIJNOBACKUP, AIJ contains no transactions that qualify for backup
%RMU−I−OPERNOTIFY, system operator notification: AIJ backup operation completed
MALIBU>SHOW SYMBOL $STATUS
  $STATUS == "%X12C8ADAC"
```

This problem has been corrected in Oracle Rdb Release 7.3.1.1.

3.3.5 Unexpected Failure of RMU/SET AIP and RMU/SHOW AIP

Bug 17722306

In prior releases of Oracle Rdb, the RENAME TABLE and RENAME INDEX commands modified the logical area names for the referenced object, including padding spaces. This did not affect the functioning of the tables or indices but caused some confusion when using tools such as RMU/SET AIP or RMU/SHOW

AIP. For these commands, the trailing spaces are significant and they would fail to find the named objects.

The following example shows the renaming of a table. Using SET FLAGS 'STOMAP_STATS' you can clearly see the log messages using space padded names.

```
SQL> set flags 'stomap_stats';
SQL>
SQL> rename table EXAMPLE_TABLE_NAME
cont>     to NEW_EXAMPLE_TABLE_NAME;
~As: reads: async 0 synch 53, writes: async 20 synch 0
SQL>
SQL> commit;
%RDMS-I-LOGMODSTR,     modified logical area name to "NEW_EXAMPLE_TABLE_NAME
      "
%RDMS-I-LOGMODSTR,     modified logical area name to "NEW_EXAMPLE_TABLE_NAME
      "
%RDMS-I-LOGMODSTR,     modified logical area name to "NEW_EXAMPLE_TABLE_NAME
      "
%RDMS-I-LOGMODSTR,     modified logical area name to "NEW_EXAMPLE_TABLE_NAME
      "
SQL>
```

A subsequent attempt to update the logical areas (using RMU/SET AIP) fails to locate the new name, as is true for RMU/SHOW AIP.

```
$ rmu/set aip abc NEW_EXAMPLE_TABLE_NAME/length/log
%RMU-F-NOLAREAFOUND, No logical areas match the specified selection parameters
%RMU-F-FTL_RMU, Fatal error for RMU operation at  5-NOV-2013 14:35:39.05
$
$ rmu/show aip abc NEW_EXAMPLE_TABLE_NAME
*------------------------------------------------------------------------------
* Logical Area Name             LArea PArea   Len Type
*------------------------------------------------------------------------------
$
```

However, using a wildcard, the names are located.

```
$ rmu/show aip abc NEW_EXAMPLE_TABLE_NAME*/brief
*------------------------------------------------------------------------------
* Logical Area Name             LArea PArea   Len Type
*------------------------------------------------------------------------------
NEW_EXAMPLE_TABLE_NAME             61     3   665 TABLE
NEW_EXAMPLE_TABLE_NAME             62     4   665 TABLE
NEW_EXAMPLE_TABLE_NAME             63     5   665 TABLE
NEW_EXAMPLE_TABLE_NAME             64     2   665 TABLE
$
```

The problem with RMU/SET AIP and RMU/SHOW AIP can be avoided by using wildcard specifications of the object name, using "*" wildcard to match the trailing spaces, or using delimited names on the DCL command line that include the trailing spaces.

The database administrator can modify the names of the logical areas using RMU/SET AIP/RENAME. However, Oracle suggests great care be taken to not change the name (apart from removing trailing spaces) as this will cause issues with future DDL operations.

```
$ rmu/set aip abc NEW_EXAMPLE_TABLE_NAME*/rename=NEW_EXAMPLE_TABLE_NAME/log
%RMU-I-AIPSELMOD, Logical area id 61, name NEW_EXAMPLE_TABLE_NAME
selected for modification
```

```
%RMU-I-AIPSELMOD, Logical area id 62, name NEW_EXAMPLE_TABLE_NAME
selected for modification
%RMU-I-AIPSELMOD, Logical area id 63, name NEW_EXAMPLE_TABLE_NAME
selected for modification
%RMU-I-AIPSELMOD, Logical area id 64, name NEW_EXAMPLE_TABLE_NAME
selected for modification
$
$ rmu/show aip abc NEW_EXAMPLE_TABLE_NAME/brief
*------------------------------------------------------------------------------
* Logical Area Name             LArea PArea   Len Type
*------------------------------------------------------------------------------
NEW_EXAMPLE_TABLE_NAME             61     3   665 TABLE
NEW_EXAMPLE_TABLE_NAME             62     4   665 TABLE
NEW_EXAMPLE_TABLE_NAME             63     5   665 TABLE
NEW_EXAMPLE_TABLE_NAME             64     2   665 TABLE
$
```

This problem has been corrected in Oracle Rdb Release 7.3.1.1. Oracle Rdb has been corrected to pass the object name without the padding spaces.

3.3.6 RMU/SET AUDIT Ignoring "*" Wildcard for the IDENTIFIERS Option

In prior releases of Oracle Rdb, the RMU Set Audit command ignored the "*" when specified for /ENABLE=IDENTIFIERS=* or /DISABLE=IDENTIFIERS=* to select all users to be audited.

The following example shows that a successful RMU command didn't change the enabled identifiers.

```
$ rmu/show audit/ident TEST_DB

Enabled identifiers:
    None

$ rmu/set audit/enable=ident=* TEST_DB
$ rmu/show audit/ident TEST_DB

Enabled identifiers:
    None

$
```

A workaround to this problem is to specify the string "[*]" which is an equivalent specification.

This problem has been corrected in Oracle Rdb Release 7.3.1.1. RMU now correctly interprets the "*" as a request to audit all user and role (rights) identifiers used by the database.

```
$ rmu/show audit/ident TEST_DB

Enabled identifiers:
    None

$ rmu/set audit/enable=ident=* TEST_DB
$ rmu/show audit/ident TEST_DB

Enabled identifiers:
    (IDENTIFIER=*)
```

```
$
```

Chapter 4
Software Errors Fixed in Oracle Rdb Release 7.3.1.0

This chapter describes software errors that are fixed by Oracle Rdb Release 7.3.1.0.

4.1 Software Errors Fixed That Apply to All Interfaces

4.1.1 Make Values in RDB$CLIENT_DEFAULTS.DAT Case Insensitive

Bug 7681548

In previous versions of Oracle Rdb, the values for the parameters SQL_MESSAGE_VECTOR_RETURN_TYPE, SQL_NETWORK_TRANSPORT_TYPE and SQL_DEFAULTS_RESTRICTION in the configuration file RDB$CLIENT_DEFAULTS.DAT only took effect if specified in UPPER CASE. For example, if decnet was specified as the transport type, then the default (to first try DECnet and then TCPIP) would still be applied. Only if DECNET was specified in all UPPER CASE would only DECnet be used.

This problem has been corrected in Oracle Rdb Release 7.3.1.0. The values for SQL_MESSAGE_VECTOR_RETURN_TYPE, SQL_NETWORK_TRANSPORT_TYPE and SQL_DEFAULTS_RESTRICTION can now be specified in UPPER, lower or Mixed case with the same result in all three cases.

Note that for systems where, before an upgrade to Oracle Rdb Release 7.3.1, SQL_NETWORK_TRANSPORT_TYPE was specified as "decnet" with one or more lower case characters in this string, the behavior might change after an upgrade. Such systems will now strictly use DECnet as the transport while before the upgrade they would use the default which is to attempt to connect via TCPIP if the attempt to connect via DECnet failed.

4.1.2 Query Ignores Potentially Useful BgrNdx

Bug 2586052

The Dynamic optimizer ignores some potentially useful index as background index (BgrNdx) that could improve the performance at run time.

For example, the following query shows that the index I23 could be considered as a useful background index:

```
show table (index) t2
Indexes on table T2:
I21                             with column F1
                                and column F2
...etc...

Indexes on table T2:
I22                             with column F1
                                and column F3
...etc...

Indexes on table T2:
I23                             with column F2
                                and column F3
```

```
...etc...

select count(*) from t2 where f1 = 1 and f2 = 1 and f3 = 1;
Tables:
  0 = T2
Aggregate: 0:COUNT (*) Q2
Leaf#01 BgrOnly 0:T2 Card=64
  Bool: (0.F1 = 1) AND (0.F2 = 1) AND (0.F3 = 1)
  BgrNdx1 I21 [2:2] Fan=14
    Keys: (0.F1 = 1) AND (0.F2 = 1)
  BgrNdx2 I22 [2:2] Fan=14
    Keys: (0.F1 = 1) AND (0.F3 = 1)
~Estim  I21 Sorted: Split lev=1, Seps=1 Est=4
~Estim  I22 Sorted: Split lev=1, Seps=1 Est=4
~E#0011.01(1) Estim   Index/Estimate 1/4 2/4
~E#0011.01(1) BgrNdx1 EofData  DBKeys=4  Fetches=0+0  RecsOut=0 #Bufs=1
~E#0011.01(1) BgrNdx2 FtchLim  DBKeys=0  Fetches=0+0  RecsOut=0
~E#0011.01(1) Fin     Buf      DBKeys=4  Fetches=0+1  RecsOut=1

           1
1 row selected
```

Note that I23 is not used as BgrNdx3 in this case since we already had all the DBKEYs covered by BgrNdx1 for F1 and F2 and BgrNdx2 for F3. The following query shows the obvious problem with performance when I23 is ignored as BgrNdx3.

```
select count(*) from t2
     where (f1 = 1 or (f1 between 1 and 100))
     and f2 > 80
     and f3 < 100;
Tables:
  0 = T2
Aggregate: 0:COUNT (*) Q2
Leaf#01 BgrOnly 0:T2 Card=64
  Bool: ((0.F1 = 1) OR ((0.F1 >= 1) AND (0.F1 <= 100))) AND (0.F2 > 80) AND (
        0.F3 < 100)
  BgrNdx1 I21 [1:1,2:1] Fan=14
    Keys: r0: (0.F1 >= 1) AND (0.F1 <= 100)
          r1: (0.F1 = 1) AND (0.F2 > 80)
    Bool: 0.F2 > 80
  BgrNdx2 I22 [1:1,1:2] Fan=14
    Keys: r0: (0.F1 >= 1) AND (0.F1 <= 100)
          r1: (0.F1 = 1) AND (0.F3 < 100)
    Bool: 0.F3 < 100
~Estim I21 Sorted: Split lev=1, Seps=16 Est=64
~Estim I22 Sorted: Split lev=1, Seps=16 Est=64
~E#0012.01(1) Estim   Index/Estimate 1/64 2/64
~E#0012.01(1) BgrNdx1 EofData  DBKeys=16  Fetches=0+0  RecsOut=0 #Bufs=1
~E#0012.01(1) BgrNdx2 FtchLim  DBKeys=0  Fetches=0+0  RecsOut=0
~E#0012.01(1) Fin     Buf      DBKeys=16  Fetches=0+0  RecsOut=16

          16
1 row selected
```

In this case, I23 would be the perfect index since it covers F2 and F3, and should be considered as BgrNdx3.

This problem has been corrected in Oracle Rdb Release 7.3.1.0.

4.1.3 Query Runs Slow Executing BGRNDX2 With Full Index Scan

Bugs 4731889 and 4020688

In Bug 4731889, the following query spent a significant amount of time (approximately 26 CPU seconds as compared to less than 1 CPU second) fetching over 80k IO's in the BGRNDX2 without finding useful matching records.

```
Select * from CTR_TAB
where
    CTR_FID='129' and
    CTR_BDATE>=20040712 and
    CTR_STATUS='1' and
    ((CTR_ADATE>=20040712) or
     (CTR_ADATE=20040712 and CTR_ATIME>=15300898));
Tables:
  0 = CTR_TAB
Leaf#01 FFirst 0:CTR_TAB Card=17244926
  Bool: (0.CTR_FID = '129') AND (0.CTR_BDATE >= 20040712) AND (
        0.CTR_STATUS = '1') AND ((0.CTR_ADATE >= 20040712) OR ((
        0.CTR_ADATE = 20040712) AND (0.CTR_ATIME >= 15300898)))
  BgrNdx1 CTR_FID_NDX [2:1,2:2] Fan=25
    Keys: r0: (0.CTR_FID = '129') AND (0.CTR_ADATE = 20040712)
          r1: (0.CTR_FID = '129') AND (0.CTR_ADATE >= 20040712)
    Bool: 0.CTR_FID = '129'
  BgrNdx2 CTR_UNIQUE_NDX [0:0] Fan=14
    Bool: (0.CTR_FID = '129') AND (0.CTR_BDATE >= 20040712) AND ((
          0.CTR_ADATE >= 20040712) OR ((0.CTR_ADATE = 20040712) AND
          (0.CTR_ATIME >= 15300898)))
~Estim  CTR_FID_NDX Sorted: Split lev=3, Seps=4 Est=3345
~Estim  CTR_UNIQUE_NDX Sorted: Split lev=6, Seps=2 Est=2345530
~E#0001.01(1) Estim   Index/Estimate 1/3345 2/2345530
~E#0004.01(1) BgrNdx1 EofBuf   DBKeys=1024  Fetches=2+19  RecsOut=0
~E#0004.01(1) BgrNdx2 EofBuf   DBKeys=1024  Fetches=4+80967  RecsOut=0
~E#0004.01(1) BgrNdx1 EofBuf   DBKeys=2048* Fetches=0+17  RecsOut=0
~E#0004.01(1) BgrNdx1 EofBuf   DBKeys=3072* Fetches=0+18  RecsOut=0
~E#0004.01(1) BgrNdx1 EofBuf   DBKeys=4096* Fetches=0+17  RecsOut=0
~E#0004.01(1) BgrNdx1 EofBuf   DBKeys=5120* Fetches=0+18  RecsOut=0
~E#0004.01(1) BgrNdx1 EofBuf   DBKeys=6144* Fetches=0+17  RecsOut=0
~E#0004.01(1) BgrNdx1 EofBuf   DBKeys=7168* Fetches=0+18  RecsOut=0
~E#0004.01(1) BgrNdx1 EofBuf   DBKeys=8192* Fetches=0+19  RecsOut=0
~E#0004.01(1) BgrNdx1 EofData  DBKeys=8408* Fetches=0+4  RecsOut=0 #Bufs=6832
~E#0004.01(1) FgrNdx  FFirst   DBKeys=0  Fetches=0+7029  RecsOut=0`ABA
~E#0004.01(1) Fin     TTblIni  DBKeys=0  Fetches=0+0  RecsOut=0`ABA
0 rows selected
show stat

                  process statistics at <date-time>
         elapsed time =   0 00:05:05.50             CPU time =   0 00:00:26.16
     page fault count = 4453            pages in working set = 41184
   buffered I/O count = 97                  direct I/O count = 88328
      open file count = 13              file quota remaining = 1987
           locks held = 215                  locks remaining = 31785
      CPU utilization = 8.5%             AST quota remaining = 995
```

In Bug 4020688, the following query spent quite some time (~01:21 CPU minutes as compared to less than 1 CPU second) using full scan through the BGRNDX2 fetching over 102K IO's without finding matching rows.

```
select ggggg, dbkey from    test_tbl
where
      aaaaa               =   'M14A250221'
  and bbbbb               =   ''
  and ddddd               =   '41010'
order by eeeee desc limit to 1 rows;

Tables:
  0 = TEST_TBL
Firstn: 1
Sort: 0.EEEEE(d)
Leaf#01 BgrOnly 0:TEST_TBL Card=9379030
  Bool: (0.AAAAA = 'M14A250221') AND (0.BBBBB = '') AND (0.DDDDD = '41010')
  BgrNdx1 TEST_TBL_I1 [2:2] Fan=5
    Keys: (0.AAAAA = 'M14A250221') AND (0.BBBBB = '')
  BgrNdx2 TEST_TBL_I2 [0:0] Fan=4
    Bool: (0.AAAAA = 'M14A250221') AND (0.BBBBB = '') AND (0.DDDDD = '41010')
~Estim  TEST_TBL_I1 Sorted: Split lev=4, Seps=1 Est=204
~Estim  TEST_TBL_I2 Sorted: Split lev=7, Seps=2 Est=30324
~E#0001.01(1) Estim   Index/Estimate 1/204 2/30324
~E#0001.01(1) BgrNdx1 EofBuf   DBKeys=1024  Fetches=3+85  RecsOut=0
~E#0001.01(1) BgrNdx2 FtchLim  DBKeys=0  Fetches=5+102391  RecsOut=0
~E#0001.01(1) BgrNdx2 FtchLim  DBKeys=1  Fetches=0+51234  RecsOut=0
~E#0001.01(1) Fin     Seq      DBKeys=9379030  Fetches=0+108154  RecsOut=1129
 GGGGG                               DBKEY
                              58:3640066:6
1 row selected
show stat

                  process statistics at <date−time>
         elapsed time =   0 00:04:52.64             CPU time =   0 00:01:20.87
     page fault count = 36150           pages in working set = 104208
   buffered I/O count = 67                  direct I/O count = 261978
      open file count = 10              file quota remaining = 1990
           locks held = 162                  locks remaining = 31838
      CPU utilization = 27.6%             AST quota remaining = 994
```

This problem has been corrected in Oracle Rdb Release 7.3.1.0.

4.1.4 Filter Predicates are Ignored in Aggregate Query

Bug 14133515

When COUNT (column−name) can use a SORTED RANKED index, the optimizer might select "Index Counts Lookup" strategy but it ignores the implicit FILTER (column−name is not null) that should be applied before the COUNT.

The following example shows the problem. The COUNT query returns correct results (64 rows) using a SORTED index.

```
SQL> show index mi_index
Indexes on table EMPLOYEES:
MI_INDEX                        with column MIDDLE_INITIAL
  Duplicates are allowed
  Type is Sorted
  Key suffix compression is DISABLED
SQL> select count(middle_initial) from employees;
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (0.MIDDLE_INITIAL) Q2
Index only retrieval of relation 0:EMPLOYEES
  Index name  MI_INDEX [0:0]

          64
1 row selected
```

However, it returns 100 rows (incorrectly) when SORTED RANKED index is used, as in the following example.

```
SQL> show index mi_index
Indexes on table EMPLOYEES:
MI_INDEX                        with column MIDDLE_INITIAL
  Duplicates are allowed
  Type is Sorted Ranked
    Duplicates are Compressed Bitmaps
  Key suffix compression is DISABLED
  Node size 430

SQL> select count(middle_initial) from employees;
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (0.MIDDLE_INITIAL) Q2
Index only retrieval of relation 0:EMPLOYEES
  Index name  MI_INDEX [0:0]     Index counts lookup

         100
1 row selected
```

Here is an example of a related problem when an explicit FILTER clause is ignored.

```
! the following should return 3 rows
SQL> select middle_initial from employees where middle_initial = 'R';
Tables:
  0 = EMPLOYEES
Index only retrieval of relation 0:EMPLOYEES
  Index name  MI_INDEX [1:1]
    Keys: 0.MIDDLE_INITIAL = 'R'
 MIDDLE_INITIAL
 R
 R
 R
3 rows selected

SQL> select count(middle_initial) from employees where middle_initial = 'R';
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (0.MIDDLE_INITIAL) Q2
Index only retrieval of relation 0:EMPLOYEES
  Index name  MI_INDEX [1:1]     Index counts lookup
    Keys: 0.MIDDLE_INITIAL = 'R'

           3
1 row selected
```

However, it returns the wrong result (100 rows) if the predicate is applied as a filter.

```
SQL> select count(middle_initial) filter (where middle_initial = 'R') from
EMPLOYEES;
```

```
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (0.MIDDLE_INITIAL) Q2
             Bool: 0.MIDDLE_INITIAL = 'R'
Index only retrieval of relation 0:EMPLOYEES
  Index name  MI_INDEX [0:0]     Index counts lookup

         100
1 row selected
```

A similar query using either MAX or MIN aggregate function fails using either SORTED or SORTED RANKED index.

```
SQL> select max(middle_initial) filter (where middle_initial = 'R') from
EMPLOYEES;
Tables:
  0 = EMPLOYEES
Aggregate: 0:MAX (0.MIDDLE_INITIAL) Q2
             Bool: 0.MIDDLE_INITIAL = 'R'
Index only retrieval of relation 0:EMPLOYEES
  Index name  MI_INDEX [0:0]     Max key lookup

 NULL
1 row selected

! create a sorted index
!
SQL> show index MI_INDEX
Indexes on table EMPLOYEES:
MI_INDEX                        with column MIDDLE_INITIAL
  Duplicates are allowed
  Type is Sorted
  Key suffix compression is DISABLED

SQL> select max(middle_initial) filter (where middle_initial = 'R') from
EMPLOYEES;
Tables:
  0 = EMPLOYEES
Aggregate: 0:MAX (0.MIDDLE_INITIAL) Q2
             Bool: 0.MIDDLE_INITIAL = 'R'
Index only retrieval of relation 0:EMPLOYEES
  Index name  MI_INDEX [0:0]     Max key lookup

 NULL
1 row selected
```

These problems have been corrected in Oracle Rdb Release 7.3.1.0.

4.1.5 Parallel Index Build Name Restriction Relaxed

Prior Oracle Rdb releases included the following documented restriction related to index names longer than 27 characters used when performing parallel index builds:

- For effective parallel sort index builds against the same database table, the index name of each index being built concurrently must be unique within the first 27 characters. Failure to specify a unique name creates only one sort index because each index build requests the same name lock prior to the start of each index build.

Attempts to perform parallel index builds with indexes that were not unique within the first 27 characters of the index names could result in misleading error SQL−F−IND_EXISTS messages. The following example demonstrates the create operations for EMPLOYEE_APPLICATION_CREDIT_1 and EMPLOYEE_APPLICATION_CREDIT_2 (these names are 29 characters long) running at the same time:

```
In session 1:

  SQL> CREATE INDEX EMPLOYEE_APPLICATION_CREDIT_1

In session 2:

  SQL> CREATE INDEX EMPLOYEE_APPLICATION_CREDIT_2
    %SQL−F−IND_EXISTS, Index EMPLOYEE_APPLICATION_CREDIT_2 already exists in
    this database or schema
```

This documented restriction has been relaxed and the problem corrected in Oracle Rdb Release 7.3.1.0. The entire name of the index is now used for the lock, allowing parallel index builds even when the first 27 characters of the index names are not unique.

4.1.6 EXQUOTA Caused Inaccessible AIJ

Bug 5120555

In prior versions of Oracle Rdb, if any After Image Journal (AIJ) write operation failed with a FILACCERR error and COMMIT TO JOURNAL was enabled, the AIJ file would be immediately marked as inaccessible, the database would be shutdown, and the offending process (database user or database server) would be terminated. Manual intervention would be required to reset journalling before the database could be re−opened.

Starting in this release, Oracle Rdb has eased one of the restrictions. Now, if a user process gets an FILACCERR error due to "exceeded quota" (EXQUOTA), the AIJ will not be marked as inaccessible and the database won't be shutdown. The process will, however, still be terminated. This abnormal termination will cause a Database Recovery (DBR) process to be automatically started to recover that user. No manual intervention will be required and normal database processing will continue after the DBR completes.

Note that this change will affect only database user processes. The behavior of database server processes (such as ALS, RCS, LCS, LRS) remains the same.

This problem has been corrected in Oracle Rdb Release 7.3.1.0.

4.1.7 Query Bugchecks with MAX, MIN or COUNT

Bug 5245269

When the dialect is set to enable NULL elimination warnings, a select query with either MAX, MIN or COUNT using a partitioned index, may bugcheck.

The following is an example of the problem stated above:

```
$ SQL$
SQL> attach 'filename PERSONNEL2';
SQL> set flags 'strategy,detail(2)';
SQL> create index MI_INDEX
cont>     on EMPLOYEES (MIDDLE_INITIAL)
cont>     type is SORTED RANKED
cont>     store using (MIDDLE_INITIAL)
cont>         in AREA_20 with limit of ('F')
cont>         in AREA_21 with limit of ('Z')
cont>         otherwise in AREA_22
cont> ;
SQL>
SQL> select max(middle_initial) from employees;
Tables:
  0 = EMPLOYEES
Aggregate: 0:MAX (0.MIDDLE_INITIAL) Q2
Index only retrieval of relation 0:EMPLOYEES
  Index name  MI_INDEX [0:0]     Max key lookup
%RDMS−I−BUGCHKDMP, generating bugcheck dump file USD04:[JONES]RDSBUGCHK.DMP;
```

The query works without the partitioned index defined, as in the following example:

```
SQL> drop index MI_INDEX;
SQL> select max(middle_initial) from employees;
Tables:
  0 = EMPLOYEES
Aggregate: 0:MAX (0.MIDDLE_INITIAL) Q2
Get     Retrieval sequentially of relation 0:EMPLOYEES

 Z
1 row selected
%RDB−I−ELIM_NULL, null value eliminated in set function
```

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

1. The SQL dialect is set to SQL92, SQL99, ORACLE LEVEL1, or ORACLE LEVEL2.
2. The index is partitioned.
3. The function is either MAX (where the "Max key lookup" strategy is employed), MIN (where the "Min key lookup" strategy is employed) or COUNT (where the "Index counts lookup" or "Index distinct lookup" strategy is employed).

This problem has been corrected in Oracle Rdb Release 7.3.1.0.

4.2 SQL Errors Fixed

4.2.1 NULL Elimination Semantics Now Supported by COUNT Function

In prior versions of Oracle Rdb, the COUNT aggregate function did not support the ANSI and ISO SQL Language Standard NULL Elimination semantics. This support should report the warning "null value eliminated in set function" whenever the row set included a NULL value that was ignored when computing the count of the valueexpression. This problem effected both COUNT (valueexpression) and COUNT (DISTINCT valueexpression).

In prior versions, these functions were implicitly defined as COUNT (*) FILTER (WHERE valueexpression IS NOT NULL) and so NULL values were never processed by the COUNT function. This can be seen in the strategy display shown in the following example.

```
SQL> select count(middle_initial)
cont> from employees;
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (0.MIDDLE_INITIAL) Q2
             Bool: NOT MISSING (0.MIDDLE_INITIAL)
Get     Retrieval sequentially of relation 0:EMPLOYEES

          64
1 row selected
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.1.0. Please note that query encoding for COUNT (valueexpression) and COUNT (DISTINCT valueexpression) has changed in this release and any query outlines defined for queries, views or procedures using these functions will need to be recreated.

The following example shows the new strategy and results.

```
SQL> select count(middle_initial)
cont> from employees;
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (0.MIDDLE_INITIAL) Q2
Index only retrieval of relation 0:EMPLOYEES
  Index name  MI_INDEX [0:0]

                   64
1 row selected
%RDB−I−ELIM_NULL, null value eliminated in set function
SQL>
```

4.2.2 Unexpected FOREIGN KEY Constraint Failure Due to Mismatched Evaluating Time

Bug 3858486

In prior releases of Oracle Rdb, a PRIMARY KEY or UNIQUE constraint could be defined with an evaluating time of DEFERRABLE and subsequently referenced by a FOREIGN KEY constraint that was defined as NOT DEFERRABLE. In such cases, it was possible for the PRIMARY KEY or UNIQUE columns to violate the constraint (just a transient state until a COMMIT or SET CONSTRAINT ALL statement was executed). This transient state of the data could cause the FOREIGN KEY constraint to fail unexpectedly.

In this release of Oracle Rdb, SQL will enforce the SQL Standard semantics such that the PRIMARY KEY or UNIQUE constraint must be evaluated at the same time or sooner than the referencing FOREIGN KEY constraints. It is possible that existing customer databases contain such definitions therefore only a warning will be generated for most dialects. If the DIALECT is set to SQL2011 or ORACLE LEVEL3, an error will be raised.

The following example shows the error reported when SQL2011 dialect is selected.

```
create table DEPARTMENTS (
    DEPARTMENT_CODE
        CHAR (4),
    DEPARTMENT_NAME
        CHAR (30),
    MANAGER_ID
        CHAR (5)
        constraint DEPT_EMPID_FK
            references EMPLOYEES (EMPLOYEE_ID)
            not deferrable,
    BUDGET_PROJECTED
        INTEGER,
    BUDGET_ACTUAL
        INTEGER);
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−E−UNQCONFEVAL, the constraint "EMPL_EMPID_PK" referenced by
"DEPT_EMPID_FK" has a conflicting evaluation time attribute
```

This problem has been corrected in Oracle Rdb Release 7.3.1.0.

4.2.3 Unexpected RDB−E−OBSOLETE_METADA Error During ALTER TABLE

Bug 17361878

In prior releases of Oracle Rdb, the implicit alter of a view definition was not handling missing metadata items correctly. When a COMPUTED BY column of a table or view cannot be resolved because of a prior DROP ... CASCADE operation, the value returned for the computed expression should be NULL. The database administrator should replace the missing object as soon as possible.

The reported problem shows that subsequent ALTER TABLE operations fail when the missing column is referenced. This is shown by this simple example.

```
SQL> create table TEST_TBL
cont>     (c0     bigint
cont>     ,c1     tinyint
cont>     );
SQL>
SQL> insert into TEST_TBL
cont>     values (0, 1);
```

```
1 row inserted
SQL>
SQL> alter table TEST_TBL
cont>     add c2
cont>     automatic as −1;
SQL>
SQL> create view TEST_TBL_VIEW
cont>     as select * from TEST_TBL;
SQL>
SQL> alter table TEST_TBL
cont>     drop column c2 cascade;
SQL>
SQL> alter table TEST_TBL
cont>     alter C1 smallint;
%RDB−E−NO_META_UPDATE, metadata update failed
−RDB−E−OBSOLETE_METADA, request references metadata objects that no longer exist
−RDMS−F−BAD_SYM, unknown field symbol − C2
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.1.0.

4.2.4 Unexpected RDMS−F−ACTQUERY Query Error From ALTER TABLE ... DROP COLUMN

In prior versions of Oracle Rdb, attempts to use ALTER TABLE to drop a column would fail with a RDMS−F−ACTQUERY error. This occurred because the table being altered was in use by a compile query, such as a declared cursor. This is normal for the RESTRICT option (the default behavior) but should not happen when using the CASCADE option. When using DROP TABLE ... CASCADE no such error occurs because Oracle Rdb implicitly invalidates active queries.

The following example shows the reported error.

```
SQL> DECLARE C11332 CURSOR FOR SELECT COUNT(*) FROM CHANGG;
SQL> OPEN C11332;
SQL> DECLARE :INT1 INT;
SQL> FETCH C11332 INTO :INT1;
SQL> CLOSE C11332;
SQL> ALTER TABLE CHANGG DROP AGE CASCADE;
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−F−ACTQUERY, there are queries compiled that reference relation "CHANGG"
−RDMS−F−RELNOTCHG, relation CHANGG has not been changed
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.3.1.0. When the CASCADE option is used, the ALTER TABLE statement invalidates active queries in the same way as the DROP TABLE ... CASCADE statement.

4.3 RMU Errors Fixed

4.3.1 Unexpected Definitions in RMU Extract Output

Bug 14786014

In prior releases of Oracle Rdb, the RMU Extract command would include special objects created by OCI Services for Rdb. These data dictionary objects (domains, functions, modules, procedures, tables and views) are marked as hidden by Rdb and are not usually visible. However, RMU Extract was including them in the output SQL script, often making the output confusing.

The new option Hidden_Objects can be used to display these definitions if desired. The default is NoHidden_Objects.

This problem has been corrected in Oracle Rdb Release 7.3.1.0.

4.3.2 RMU BACKUP GROUP_SIZE Default Value Increased

The RMU BACKUP "GROUP_SIZE" qualifier specifies the frequency at which XOR recovery blocks are written to tape. In prior releases of Oracle Rdb, the default group size when writing to a tape device was 10.

Starting with Oracle Rdb Release 7.3.1.0, the default group size when writing to a tape device is 100. This change may result in increased throughput and a reduction in tape space used.

4.3.3 RMU Parallel Backup Fails With /PROTECTION Qualifier

In releases prior to Oracle Rdb Release 7.3.1.0, there was a problem in RMU/BACKUP/PARALLEL and RMU/BACKUP/PLAN where the information specified in the /PROTECTION qualifier was not correctly reflected in the plan file. For example, the following RMU command:

```
$ RMU/BACKUP/PARALLEL=EXECUTOR=1/DISK/EXECUTE /PROTECTION=(S:,O:,G:W,W:R) -
    /LIST_PLAN=MFP.PLAN MF_PERSONNEL DISK1:[MF_P]MFP.RBF
```

resulted in the following entry in the plan file:

```
    Protection = S:
```

Also, RMU/BACKUP/PLAN/LIST_PLAN failed with the following errors.

```
%CLI-F-SYNTAX, error parsing 'PROTECTION'
-CLI-E-ENTNF, specified entity not found in command tables
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 19-JUN-2013 15:35:05
```

Both of these problems have been corrected in Oracle Rdb Release 7.3.1.0.

4.3.4 Improvements to RMU/COLLECT OPTIMIZER_STATISTICS

This release of Oracle Rdb improves RMU/COLLECT OPTIMIZER_STATISTICS /STATISTIC=WORKLOAD in the following areas.

- In prior versions of Oracle Rdb, the existence of a UNIQUE HASHED SCATTERED or UNIQUE HASHED ORDERED index was not considered when estimating various workload statistics. This has been corrected.
- In addition, even if a UNIQUE SORTED or UNIQUE SORTED RANKED index was found, the estimated value for duplicity (RDB$DUPLICITY_FACTOR) was over estimated. Rdb knows from the index that there are no duplicates and the duplicity factor is computed as 1 / table−cardinality (or zero if the table is empty). This has now been corrected.

Oracle recommends running RMU/COLLECT OPTIMIZER_STATISTICS against your workload as soon as possible after the upgrade to Oracle Rdb Release 7.3.1.0.

4.4 RMU Show Statistics Errors Fixed

4.4.1 RMU/SHOW/STATISTICS Avoids VASFULL Errors By Moving to P2 Address Space

Bug 12921679

In releases prior to Oracle Rdb 7.3.1.0, RMU/SHOW/STATISTICS gave a VASFULL error if the largest logical area number was a large number.

The following is an example of the errors seen with RMU/SHOW/STATISTICS when the maximum logical area ID was a large number.

```
$RMU/SHOW STATISTICS TEST_DB
%COSI−F−VASFULL, virtual address space full
−SYSTEM−F−ILLPAGCNT, illegal page count parameter
%RMU−F−FATALOSI, Fatal error from the Operating System Interface.
%RMU−F−FTL_SHOW, Fatal error for SHOW operation at 26−AUG−2012 10:53:36.75
$ SQL
SQL>ATTACH 'FILENAME TEST_DB';
SQL>SELECT MAX(RDB$LOGICAL_AREA_ID) FROM RDB$LOGICAL_AREAS;

         21987
1 row selected
```

This problem has been corrected in Oracle Rdb Release 7.3.1.0. This release eases the memory restriction by making use of P2 address space. The COSI−F−VASFULL error should no longer occur.

Chapter 5
Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.2

5.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.2

5.1.1 New FULBCKREQ Message Output When a Full Backup is Required

Bug 18328148

There are some Oracle Rdb database changes that require the next database backup to be a full backup to guarantee correct database recovery using an incremental backup. If an incremental database backup is executed without a preceding full database backup, the incremental backup will be aborted with a fatal error.

```
$ rmu/backup/incremental/nolog mf_personnel.rdb -
  DEVICE:[DIRECTORY]mfp.rbf
%RMU-F-NOFULLBCK, no full backup of this database exists
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 14-MAR-2014 13:45:21.35
```

A dump of the database header will show if this root flag is set.

```
$ rmu/dump/header mf_personnel
*------------------------------------------------------------------------------
* Oracle Rdb V7.3-12                                    14-MAR-2014 13:45:18.51
*
* Dump of Database header
*     Database: DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
*
*------------------------------------------------------------------------------
Database Parameters:

    Database Backup...

      - Incremental backup not allowed until full backup
```

A new warning message will now be output at the end of an ALTER DATABASE command if the ALTER DATABASE command contains one or more operations which require the next database backup to be a full database backup.

```
%RDMS-W-FULBCKREQ, The next database backup must be a full backup
```

These are the operations that require the next database backup to be a full database backup.

- Add one or more storage areas to the database.

  ```
  $ SQL$
  alter data filename mf_personnel
    add storage area new_area;
  %RDMS-W-FULBCKREQ, The next database backup must be a full backup
  $
  ```
- Delete one or more storage areas from the database.

  ```
  $ SQL$
  ```

```
alter database filename mf_personnel
  drop storage area area1
  drop storage area area2
  drop storage area area3
  drop storage area area4;
%RDMS−W−FULBCKREQ, The next database backup must be a full backup
$
```

- Reserve database entries for additional storage areas.

```
$ SQL$
alter database filename mf_personnel
  reserve 10 storage areas;
%RDMS−W−FULBCKREQ, The next database backup must be a full backup
$
```

- Modify the live storage area page allocation.

```
$ SQL$
alter database filename mf_personnel
  alter storage area jobs allocation is 2000 pages;
%RDMS−W−FULBCKREQ, The next database backup must be a full backup
$
```

- Modify the maximum number of database users.

```
$ SQL$
alter database filename mf_personnel
  number of users is 50;
%RDMS−W−FULBCKREQ, The next database backup must be a full backup
$
```

- Modify the maximum number of database cluster nodes.

```
$ SQL$
alter database filename mf_personnel
  number of cluster nodes is 4;
%RDMS−W−FULBCKREQ, The next database backup must be a full backup
```

5.1.2 New TRACE Option for EXPORT DATABASE Statement

This release of Oracle Rdb adds a TRACE option to the EXPORT DATABASE Statement. This option enables tracing of certain operations internal to EXPORT. For example, when COMPRESSION and TRACE are specified, the TRACE option causes output of the compression percentages for each table, null bit vector (NBV) and list of byte varying data.

```
SQL> export database filename personnel into pers compression trace;
** compress nbv : <CANDIDATES> too small to compress
** compress data: <CANDIDATES> input 846 output 244 deflate 72%
** compress nbv : <COLLEGES> too small to compress
** compress data: <COLLEGES> input 896 output 556 deflate 38%
** compress nbv : <DEGREES> too small to compress
** compress data: <DEGREES> input 4785 output 2268 deflate 53%
** compress nbv : <DEPARTMENTS> too small to compress
** compress data: <DEPARTMENTS> input 1222 output 750 deflate 39%
** compress data: <EMPLOYEES> input 11700 output 4559 deflate 62%
** compress nbv : <EMPLOYEES> input 1200 output 808 deflate 33%
** compress nbv : <JOBS> too small to compress
** compress data: <JOBS> input 495 output 434 deflate 13%
** compress nbv : <JOB_HISTORY> too small to compress
** compress data: <JOB_HISTORY> input 9316 output 6095 deflate 35%
** compress nbv : <RESUMES> too small to compress
** compress data: <RESUMES> too small to compress
** compress nbv : <SALARY_HISTORY> too small to compress
** compress data: <SALARY_HISTORY> input 18225 output 13695 deflate 25%
** compress nbv : <WORK_STATUS> too small to compress
** compress data: <WORK_STATUS> input 69 output 66 deflate 5%
SQL>
```

In this example, several tables and null bit vectors (NBV) can not be reduced by compression because of their small size.

5.1.3 New /NOAFTER_JOURNAL Qualifier to Disable AIJ File Creation by RMU/RECOVER

Bug 6656199

Oracle Rdb normally writes information to the after image journal (AIJ) file describing the creation of new after image journal files. There are cases where the user does not want RMU/RECOVER to process this information and recreate AIJ files, such as lack of disk space. This release of Oracle Rdb adds a new /NOAFTER_JOURNAL qualifier to the RMU/RECOVER command. If this qualifier is specified, no new Rdb AIJ files will be created by the current RMU/RECOVER command and any AIJ file deletion records for those AIJ files which were not created will also be ignored.

The syntax for this new RMU/RECOVER qualifier is as follows.

```
/[NO]AFTER_JOURNAL
```

The default if this qualifier is not specified is /AFTER_JOURNAL. Therefore, /NOAFTER_JOURNAL must be specified to ignore the creation of new AIJ files recorded in any AIJ file being recovered by the current RMU/RECOVER command.

In the following example, the creation of the new after image journal file J2.AIJ for the MF_PERSONNEL database is journaled to the current after image journal file RMU_RECOVER_4.AIJ_1. When the MF_PERSONNEL database is deleted and then restored from the MF_PERSONNEL.RBF file and then recovered from RMU_RECOVER_4.AIJ_1 using the new /NOAFTER_JOURNAL qualifier, the J2.AIJ file does not get created.

```
$!
$! Change the database to enable after image journaling to the
$! RMU_RECOVER_4.AIJ_1 AIJ file
$!
$ SQL$
      alter database filename MF_PERSONNEL
      reserve 10 journals
      journal filename disk:[directory]RMU_RECOVER_4.AIJ_1;
%RDMS−W−DOFULLBCK, full database backup should be done to ensure future recovery
      exit
$!
$! Backup the database
$!
$ RMU/BACKUP/NOLOG MF_PERSONNEL DISK:[DIRECTORY]MF_PERSONNEL.RBF
$!
```

```
$! Add a new AIJ file J2.AIJ to put a create AIJ file record in the current
$! RMU_RECOVER_4.AIJ_1 AIJ file
$!
$ SQL$
 alter database file mf_personnel
        add journal j2 filename disk:[directory]j2 allocation is 1000 blocks;
exit
$!
$! Drop the database and then restore the database from the backup RBF file
$!
$ SQL$
  drop database filename MF_PERSONNEL;
  exit
$!
$! Delete the added j2.aij file
$!
$ DELETE DISK:[DIRECTORY]J2.AIJ;*
$!
$! Restore the database
$!
$ RMU/RESTORE/NOCDD/NOLOG/NOAFTER_JOURNAL -
  /ROOT=DISK:[DIRECTORY]MF_PERSONNEL.RDB DISK:[DIRECTORY]MF_PERSONNEL.RBF
$!
$! Recover the database from RMU_RECOVER_4.AIJ_1 to show that the J2.AIJ
$! file does not get created if RMU/RECOVER/NOAFTER_JOURNAL is specified
$!
$ RMU /RECOVER /NOAFTER_JOURNAL /NOLOG /ROOT=DISK:[DIRECTORY]MF_PERSONNEL.RDB -
    DISK:[DIRECTORY]RMU_RECOVER_4.AIJ_1
%RMU-I-LOGRECDB, recovering database file DISK:[DIRECTORY]MF_PERSONNEL.RDB;1
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence number
needed will be 1
%RMU-I-AIJNOENABLED, after-image journaling has not yet been enabled
$ DIR DISK:[DIRECTORY]J2.AIJ
%DIRECT-W-NOFILES, no files found
```

5.1.4 Enhance Dumper of Merge Range List

Bug 18530761

In prior releases of Oracle Rdb, the strategy display for a query with OR predicates could be misleading when multiple range lists were merged. The following example demonstrates this problem with a query performed with IN and OR predicates to restrict values to selected ranges.

```
create table TEST_TABLE
   (a char);
create index TEST_TABLE_INDEX on TEST_TABLE (A);

select * from TEST_TABLE
where a in ('A', 'A', 'V', 'B') or a between 'C' and 'Y';
Tables:
  0 = TEST_TABLE
Conjunct: (0.A = 'A') OR (0.A = 'V') OR (0.A = 'B') OR ((0.A >= 'C') AND (0.A
           <= 'Y'))
Index only retrieval of relation 0:TEST_TABLE
  Index name  TEST_TABLE_INDEX [(1:1)4]
    Keys: r0: (0.A >= 'C') AND (0.A <= 'Y')
          r1: 0.A = 'B'
          r2: 0.A = 'V'
          r3: 0.A = 'A'
0 rows selected
```

Rdb has merged the OR ranges into a single range list ('A' .. 'Y') and eliminated duplicate ranges. However, the STRATEGY and DETAIL display do not reflect this state.

In this release, use the SET FLAGS 'MERGE_RANGE_LIST' flag in addition to STRATEGY and DETAIL to display further details.

```
! turn on the display of merge_range_list
set flags 'merge_range_list';

select * from TEST_TABLE
where a in ('A', 'A', 'V', 'B') or a between 'C' and 'Y';
Tables:
  0 = TEST_TABLE
Conjunct: (0.A = 'A') OR (0.A = 'V') OR (0.A = 'B') OR ((0.A >= 'C') AND (0.A
           <= 'Y'))
Index only retrieval of relation 0:TEST_TABLE
  Index name  TEST_TABLE_INDEX [(1:1)4]
    Keys: r0: (0.A >= 'C') AND (0.A <= 'Y')
          r1: 0.A = 'B'
          r2: 0.A = 'V'
          r3: 0.A = 'A'
  Index name  TEST_TABLE_INDEX [1:1]
    Columns: r0:{(0.A),(0.A)}
    IKeys:   r0:{('A'), ('Z')}
0 rows selected
```

Note that the upper range is encoded as a higher value internally so that the index scan is terminated correctly.

5.1.5 RMU Extract Now Extracts SYS_GET_DIAGNOSTIC Function

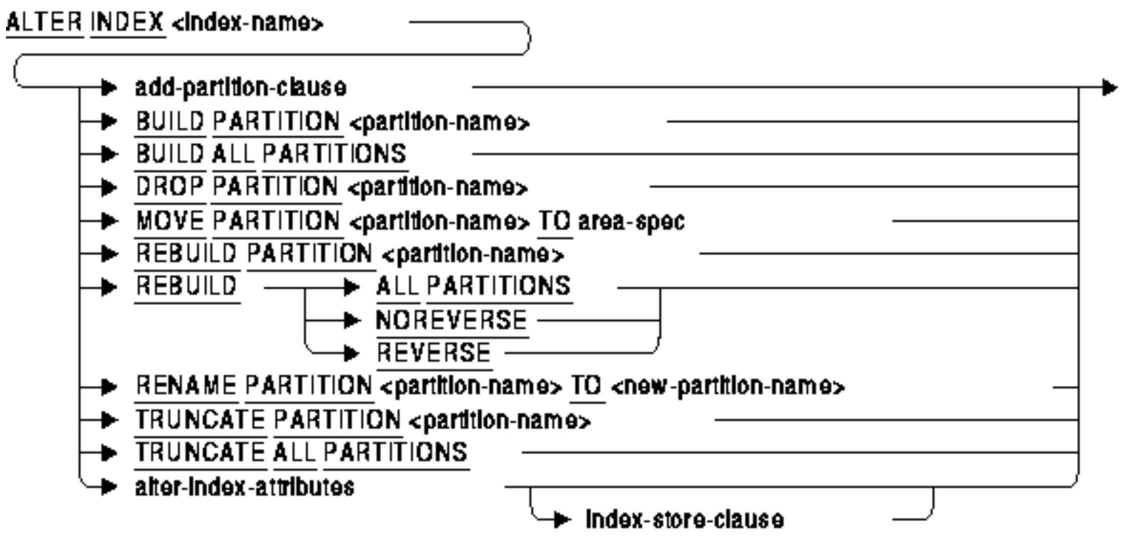
With this release of Oracle Rdb, the RMU Extract command now correctly formats the SYS_GET_DIAGNOSTIC function, which was added in earlier releases.

5.1.6 Alter Index Now Supports REVERSE and NOREVERSE Clauses

This release of Oracle Rdb now supports the REVERSE keyword on the ALTER INDEX statement as part of the REBUILD action. This clause requests that the named index be rebuilt as a REVERSE key index.

Syntax

The revised syntax for the ALTER INDEX statement is:


ALTER INDEX <index-name>
add-partition-clause
BUILD PARTITION <partition-name>
BUILD ALL PARTITIONS
DROP PARTITION <partition-name>
MOVE PARTITION <partition-name> TO area-spec
REBUILD PARTITION <partition-name>
REBUILD
ALL PARTITIONS
NOREVERSE
REVERSE
RENAME PARTITION <partition-name> TO <new-partition-name>
TRUNCATE PARTITION <partition-name>
TRUNCATE ALL PARTITIONS
alter-index-attributes
index-store-clause

REVERSE
NOREVERSE

An existing SORTED or SORTED RANKED index can be converted to a REVERSE index by using this variation of the REBUILD ALL PARTITIONS clause. An already REVERSE index can be changed to a non−REVERSE index using the NOREVERSE keyword.

The following example shows an existing index being converted to a REVERSE index.

```
SQL> alter index DOCUMENTS
cont>  rebuild reverse;
SQL>
```

Usage Notes

- If an index is already defined as REVERSE, then REBUILD REVERSE is equivalent to REBUILD ALL PARTITIONS.
- If an index is not defined as REVERSE, then REBUILD NOREVERSE is equivalent to REBUILD ALL PARTITIONS.
- The clause REVERSE may not be applied to a HASHED index.
- When applying REVERSE to an existing index, any column defined as DESC will be modified to remove the descending ordering.

5.1.7 SQL Precompiler Now Generates C++ Compatible Intermediate C Source

Enhancement Bug 1504425

This release of Oracle Rdb includes some support for using SQL Precompiler with the C++ compiler (CXX). The Rdb SQL precompiler now generates C++ compatible definitions when processing embedded SQL commands in a .SC source. This support does not support the C++ language, and the processed .SC file must conform to a legal C source format and language features.

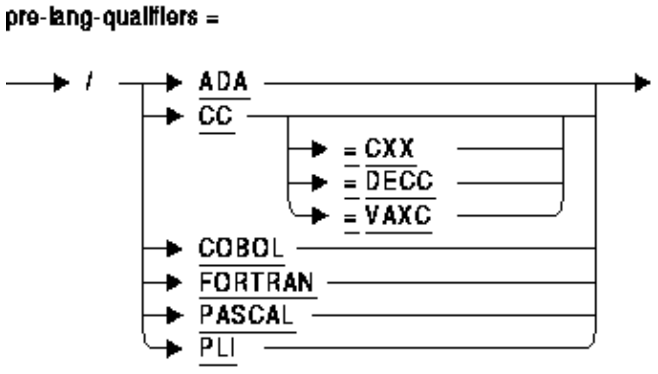
Please note the following changes:

- The SQL$PRE command line now accepts CXX as an option to the /CC qualifier. This option will direct the SQL precompiler to generate C code which is acceptable to the C++ compiler.
- The CXX DCL command will be used to invoke the C++ compiler, instead of the CC command. Additional qualifiers on the SQL$PRE command line will be passed to the CXX compiler and must be legal qualifiers for C++.
- Function prototypes will include parameter definitions
- Function prototypes are enclosed by extern "C" to prevent the names being interpreted as C++ routines.

  ```
  #ifdef __cplusplus
  extern "C" {
  #endif /* __cplusplus */
  ...
  #ifdef __cplusplus
  }
  #endif /* __cplusplus */
  ```
- SQLCODE is expected to be defined as type ***int***.
- On OpenVMS Alpha systems, the application is expected to be linked with the special library SYS$LIBRARY:LIBCXXSTD.OLB. Please refer to the relevant HP C++ documentation.

Syntax

The revised syntax for the PRE−LANG−QUALIFIERS qualifier is:



5.1.8 New RMU/RECOVER Feature to Alter Storage Areas, After_Journal and Record_Cache Directories

Enhancement Bug 867636

A new feature has been added to RMU/RECOVER and RMU/DUMP to alter the directory specifications for new storage areas, after image journal files and row caches created during the recovery of an Oracle Rdb database. This is only for the creation of new storage areas, after image journal files and row caches as recorded in the after image journal files used for a database recovery by the RMU/RECOVER command, and only for modifying the directory specifications defined in the after image journal files for the new storage areas, after image journal files and row caches at the time RMU/RECOVER creates them.

This new feature will allow the user to control where newly created storage areas, after image journal files and row caches are located. Previously, they could only be put in the directory locations recorded in the after image journal files used for the recovery.

Logicals can be used for the directory specifications but must translate to valid directory specifications that exist when the RMU/RECOVER or RMU/DUMP command is executed.

A new RMU/RECOVER /DIRECTORY qualifier can be used to specify one directory specification for all storage areas created, one directory specification for all after image journal files created and/or one directory specification for all row caches created.

The syntax for this new qualifier for storage areas is:

```
/DIRECTORY=AREAS=directory_specification
```

The syntax for this new qualifier for after image journal files is:

```
/DIRECTORY=AFTER_JOURNAL=directory_specification
```

The syntax for this new qualifier for row caches is:

```
/DIRECTORY=ROW_CACHE=directory_specification
```

To specify two or more of these options with the /DIRECTORY qualifier, use the following syntax.

```
/DIRECTORY=(AREAS=directory_specification, -
            AFTER_JOURNAL=directory_specification, -
            ROW_CACHE=directory_specification)
```

The following example shows all three options used with the /DIRECTORY qualifier in the RMU/RECOVER command.

```
$ RMU/RECOVER/NOLOG/NOTRACE/NOCONFIRM/NOAUTOMATIC-
  /DIRECTORY=(AREAS=DISK:[DIRECTORY], -
 AFTER_JOURNAL=DISK:[DIRECTORY], -
  ROW_CACHE=DISK:[DIRECTORY]) -
  DISK:[DIRECTORY]:TEST_J1_BCK.AIJ
```

To specify different directory specifications for specific storage areas, after journal files and row caches, option files can be designated using the new RMU/RECOVER /OPTIONS qualifier. One option file must be specifed to select one or more storage areas for directory modification, one option file must be specified to select one or more after image journal files for directory modification and one option file must be specified to select one or more row caches for directory modification.

The syntax for this new qualifier for storage areas is:

```
/OPTIONS=AREAS=option_file_specification
```

The syntax for this new qualifier for after image journal files is:

```
/OPTIONS=AFTER_JOURNAL=option_file_specification
```

The syntax for this new qualifier for row caches is:

```
/OPTIONS=ROW_CACHE=option_file_specification
```

To specify two or more of these options with the /OPTIONS qualifier, use the following syntax.

```
/OPTIONS=(AREAS=option_file_specification, -
          AFTER_JOURNAL=option_file_specification, -
          ROW_CACHE=option_file_specification)
```

The following example shows all three options used with the /OPTIONS qualifier in the RMU/RECOVER command.

```
$ RMU/RECOVER/NOLOG/NOTRACE/NOCONFIRM/NOAUTOMATIC-
  /OPTIONS=(AREAS=DISK:[DIRECTORY]RECOVOPTAREAS.OPT, -
  AFTER_JOURNAL=DISK:[DIRECTORY]RECOVOPTAIJ.OPT, -
  ROW_CACHE=DISK:[DIRECTORY]RECOVOPTRCACHE.OPT) -
  DISK:[DIRECTORY]TEST_J1_BCK.AIJ
```

A new /DIRECTORY_OPTIONS qualifier has been added to the RMU/DUMP command to create storage area, after image journal and row cache option files for use with the /OPTIONS qualifier in the RMU/RECOVER command. These option files will contain entries for all storage areas, after image journal files and row caches defined for the database. These option files can then be edited by the user to eliminate storage area, after image journal or row cache entries that will not be created during the RMU/RECOVER session, or they can be used without being edited since entries for storage areas, after image journal files and row caches not created during the recovery operation will be ignored.

The syntax for this new qualifier for storage areas is:

```
/DIRECTORY_OPTIONS=AREAS=option_file_specification
```

The syntax for this new qualifier for after image journal files is:

```
/DIRECTORY_OPTIONS=AFTER_JOURNAL=option_file_specification
```

The syntax for this new qualifier for row caches is:

```
/DIRECTORY_OPTIONS=ROW_CACHE=option_file_specification
```

To specify two or more of these options with the /DIRECTORY_OPTIONS qualifier, use the following syntax.

```
/DIRECTORY_OPTIONS=(AREAS=option_file_specification, -
                    AFTER_JOURNAL=option_file_specification, -
                    ROW_CACHE=option_file_specification)
```

The following example shows all three options used with the /DIRECTORY_OPTIONS qualifier in the RMU/DUMP command.

```
$ RMU/DUMP-
  /DIRECTORY_OPTIONS=(AREAS=DISK:[DIRECTORY]RECOVOPTAREAS.OPT, -
  AFTER_JOURNAL=DISK:[DIRECTORY]RECOVOPTAIJ.OPT, -
  ROW_CACHE=DISK:[DIRECTORY]RECOVOPTRCACHE.OPT) -
  DISK:[DIRECTORY]TEST
```

The following example, created by the new RMU/DUMP /DIRECTORY_OPTIONS qualifier, shows the format of the option file used with the new RMU/RECOVER /OPTIONS qualifier for storage areas. The live data storage area name is followed by /DIRECTORY=DISK:[DIRECTORY] for the live storage area file to specify the directory specification for the storage area *.RDA file. This is followed by /SNAPSHOT_DIRECTORY=DISK:[DIRECTORY] for the storage area snaphot file to specify the directory specification for the storage area *.SNP file. If /SNAPSHOT_DIRECTORY is not specified, the directory specification specified by /DIRECTORY is used for both the live and snapshot storage area files.

```
  !  Recover Areas Directory Options file for database
  !  DISK:[DIRECTORY]FILENAME.EXT;VERSION
  !  Created 22-JUL-2014 09:37:24.29
  !  Created by DUMP command

  RDB$SYSTEM -
          /directory=DISK:[DIRECTORY] -
          /snapshot_directory=DISK:[DIRECTORY]


  TEST_A1 -
          /directory=DISK:[DIRECTORY] -
          /snapshot_directory=DISK:[DIRECTORY]


  TEST_A2 -
          /directory=DISK:[DIRECTORY] -
          /snapshot_directory=DISK:[DIRECTORY]


  TEST_A3 -
          /directory=DISK:[DIRECTORY] -
          /snapshot_directory=DISK:[DIRECTORY]


  TEST_A4 -
          /directory=DISK:[DIRECTORY] -
          /snapshot_directory=DISK:[DIRECTORY]
```

The following example, created by the new /DIRECTORY_OPTIONS qualifier of the RMU/DUMP command, shows the format of the option file used with the new RMU/RECOVER /OPTIONS qualifier for after image journal files. The after image journal file name is followed by /DIRECTORY=DISK:[DIRECTORY] to specify the after image journal file directory specification. This is followed by /BACKUP_DIRECTORY=DISK:[DIRECTORY] to specify the directory specification for the after image journal backup file. If /BACKUP_DIRECTORY is not specified, the directory specification specified by /DIRECTORY is used for both the after image journal file and the after image journal backup file. If no backup directory is defined for the after image journal entry in the database, the backup directory specification will be ignored.

```
  !  Recover After Journal Options file for database
```

```
!  DISK:[DIRECTORY]FILENAME.EXT;VERSION
!  Created 22-JUL-2014 09:37:24.29
!  Created by DUMP command


TEST_J1 -
        /directory=DISK:[DIRECTORY] -
        /backup_directory=DISK:[DIRECTORY]


TEST_J2 -
        /directory=DISK:[DIRECTORY] -
        /backup_directory=DISK:[DIRECTORY]


TEST_J3 -
        /directory=DISK:[DIRECTORY] -
        /backup_directory=DISK:[DIRECTORY]


TEST_J4 -
        /directory=DISK:[DIRECTORY] -
        /backup_directory=DISK:[DIRECTORY]
```

The following example, created by the new /DIRECTORY_OPTIONS qualifier of the RMU/DUMP command, shows the format of the option file used with the new RMU/RECOVER /OPTIONS qualifier for row caches. The row cache name is followed by /DIRECTORY=DISK:[DIRECTORY] to specify the row cache directory specification. If no directory specification is defined for the row cache entry in the database, the directory specification will be ignored.

```
!  Recover Row Cache Directory Options file for database
!  DISK:[DIRECTORY]FILENAME.EXT;VERSION
!  Created 22-JUL-2014 09:37:24.29
!  Created by DUMP command


TEST_A1 -
        /directory=DISK:[DIRECTORY]


TEST_A2 -
        /directory=DISK:[DIRECTORY]


TEST_A3 -
        /directory=DISK:[DIRECTORY]


TEST_A4 -
        /directory=DISK:[DIRECTORY]
```

In the following example for database DISK:[HOME]TEST.RDB, the first RMU/RECOVER command uses the /DIRECTORY qualifier to change all directories from DISK:[HOME] to DISK:[NEW] for the storage areas TEST_A3 and TEST_A4, the after image journal files TEST_J3 and TEST_J4 and the row caches TEST_A3 and TEST_A4, whose creation was recorded in the journal file TEST_J1_BCK.AIJ. Then, the new /DIRECTORY_OPTIONS qualifier in the RMU/DUMP command is used to create option files for all TEST database after image journal files, storage areas and row caches, including those with directories changed to DISK:[NEW]. The TEST database is then deleted and again restored with all directories again set to DISK:[HOME]. The second RMU/RECOVER command then uses the /OPTIONS qualifier to specify the option files created by RMU/DUMP, which include the after image journal file, storage area and row cache directories changed by the first RMU/RECOVER to DISK:[NEW], to change the directories back again from the journaled directory specification of DISK:[HOME] to DISK:[NEW].

```
$ ! Create the TEST database
$
$ SQL
  CREATE DATABASE FILENAME DISK:[HOME]TEST
  NUMBER OF CLUSTER NODES 1
  RESERVE 6 STORAGE AREAS
  RESERVE 6 JOURNALS
  RESERVE 6 CACHE SLOTS
  ROW CACHE IS ENABLED
  CREATE STORAGE AREA RDB$SYSTEM FILENAME DISK:[HOME]TEST_SYS
  CREATE STORAGE AREA TEST_A1 FILENAME DISK:[HOME]TEST_A1
  CREATE STORAGE AREA TEST_A2 FILENAME DISK:[HOME]TEST_A2
  CREATE CACHE TEST_A1
    CACHE SIZE 100 ROWS ROW LENGTH 100 BYTES LOCATION IS 'DISK:[HOME]'
  CREATE CACHE TEST_A2
    CACHE SIZE 200 ROWS ROW LENGTH 200 BYTES LOCATION IS 'DISK:[HOME]';
  DISCONNECT ALL;

  ALTER DATABASE FILENAME TEST
    ADD JOURNAL TEST_J1 FILENAME DISK:[HOME]TEST_J1.AIJ
    BACKUP FILENAME DISK:[HOME]TEST_J1_BCK.AIJ
    ADD JOURNAL TEST_J2 FILENAME DISK:[HOME]TEST_J2.AIJ
    BACKUP FILENAME DISK:[HOME]TEST_J2_BCK.AIJ
    JOURNAL IS ENABLED
      (FAST COMMIT ENABLED);
%RDMS−W−DOFULLBCK, full database backup should be done to ensure future recovery
  EXIT;
$
$ ! Back up the original database configuration.
$
$ RMU/BACKUP/NOLOG DISK:[HOME]TEST DISK:[HOME]TEST
$
$ ! Add new journals, row caches and storage areas
$
$ SQL
  ALTER DATABASE FILENAME DISK:[HOME]TEST
    ADD JOURNAL TEST_J3 FILENAME DISK:[HOME]TEST_J3.AIJ
    BACKUP FILENAME DISK:[HOME]TEST_J3_BCK.AIJ
    ADD JOURNAL TEST_J4 FILENAME DISK:[HOME]TEST_J4.AIJ
    BACKUP FILENAME DISK:[HOME]TEST_J4_BCK.AIJ
    ADD CACHE TEST_A3
        CACHE SIZE 100 ROWS ROW LENGTH 100 BYTES LOCATION IS 'DISK:[HOME]'
    ADD CACHE TEST_A4
        CACHE SIZE 100 ROWS ROW LENGTH 100 BYTES LOCATION IS 'DISK:[HOME]'
    ADD STORAGE AREA TEST_A3 FILENAME DISK:[HOME]TEST_A3
    ADD STORAGE AREA TEST_A4 FILENAME DISK:[HOME]TEST_A4;
%RDMS−W−FULBCKREQ, The next database backup must be a full backup
  EXIT;
$
$ ! Backup TEST_J1.AIJ, where the creation of the new journals, row caches
$ ! and storage areas in DISK:[HOME] is recorded.
$
$ RMU/BACKUP/AFTER/NOLOG/NOQUIET DISK:[HOME]TEST DISK:[HOME]TEST_J1_BCK.AIJ
$
$ ! Delete the database.
$
$ SQL
```

```
  DROP DATABASE FILENAME DISK:[HOME]TEST;
  EXIT;
$
$ ! Restore the database with all storage areas, after image journals
$ ! and row caches in DISK:[HOME]
$
$ RMU/RESTORE/NOCDD/NOLOG/NORECOVER/DIR=DISK:[HOME] DISK:[HOME]TEST
%RMU−I−AIJRSTAVL, 2 after−image journals available for use
%RMU−I−AIJRSTMOD, 1 after−image journal marked as "modified"
%RMU−I−AIJISON, after−image journaling has been enabled
%RMU−W−DOFULLBCK, full database backup should be done to ensure future recovery
$
$ ! Recover the database and change the directories from DISK:[HOME] to
$ ! DISK:[NEW] for the new storage areas, after image journals and row
$ ! caches whose creation was recorded in the backed up journal file
$ ! TEST_J1_BCK.AIJ
$
$ RMU/RECOVER/NOLOG/NOTRACE/NOCONFIRM/NOAUTOMATIC−
 /DIRECTORY=(AREAS=DISK:[NEW], −
 AFTER_JOURNAL=DISK:[NEW], −
 ROW_CACHE=DISK:[NEW]) −
 DISK:[HOME]TEST_J1_BCK.AIJ
%RMU−I−LOGRECDB, recovering database file DISK:[HOME]TEST_J1_BCK.AIJ;1
%RMU−I−AIJONEDONE, AIJ file sequence 0 roll−forward operations completed
%RMU−I−AIJALLDONE, after−image journal roll−forward operations completed
%RMU−I−AIJSUCCES, database recovery completed successfully
%RMU−I−AIJFNLSEQ, to start another AIJ file recovery, the sequence number
needed will be 1
$
$ ! Create directory option files for all storage areas,
$ ! after image journals and row caches in DISK:[HOME]
$ ! or DISK:[NEW]
$
$ RMU/DUMP−
 /DIRECTORY_OPTIONS=(AREAS=DISK:[HOME]RECOVOPTAREAS.OPT, −
 AFTER_JOURNAL=DISK:HOME]RECOVOPTAIJ.OPT, −
 ROW_CACHE=DISK:[HOME]RECOVOPTRCACHE.OPT) −
 DISK:[HOME]TEST
$
$ ! Delete the database.
$
$ SQL$
  DROP DATABASE FILENAME DISK:[HOME]TEST;
  EXIT;
$
$ ! Restore the database with all storage areas, after image journals
$ ! and row caches again in DISK:[HOME]
$
$ RMU/RESTORE/NOCDD/NOLOG/NORECOVER/DIR=DISK:[HOME] DISK:[HOME]TEST
%RMU−I−AIJRSTAVL, 2 after−image journals available for use
%RMU−I−AIJRSTMOD, 2 after−image journals marked as "modified"
%RMU−F−AIJENBOVR, enabling AIJ journaling would overwrite an existing journal
%RMU−I−AIJISOFF, after−image journaling has been disabled
$
$ ! Recover the database and again change the directories from DISK:[HOME]
$ ! to DISK:[NEW] for the new storage areas, after image journals and row
$ ! caches whose creation was recorded in the backed up journal file
$ ! TEST_J1_BCK.AIJ
$
$ RMU/RECOVER/NOLOG/NOTRACE/NOCONFIRM/NOAUTOMATIC−
 /OPTIONS=(AREAS=DISK:[HOME]RECOVOPTAREAS.OPT, −
 AFTER_JOURNAL=DISK:[HOME]RECOVOPTAIJ.OPT, −
```

```
ROW_CACHE=DISK:[HOME]RECOVOPTRCACHE.OPT) -
TEST$SCRATCH:TEST_J1_BCK.AIJ
 !  Recover Areas Directory Options file for database
 !  DISK:[HOME]TEST.RDB;1
 !  Created 22-JUL-2014 09:37:24.29
 !  Created by DUMP command

 RDB$SYSTEM -
        /directory=DISK:[HOME] -
        /snapshot_directory=DISK:[HOME]


 TEST_A1 -
        /directory=DISK:[HOME] -
        /snapshot_directory=DISK:[HOME]


 TEST_A2 -
        /directory=DISK:[HOME] -
        /snapshot_directory=DISK:[HOME]


 TEST_A3 -
        /directory=DISK:[NEW] -
        /snapshot_directory=DISK:[NEW]


 TEST_A4 -
        /directory=DISK:[NEW] -
        /snapshot_directory=DISK:[NEW]
 !  Recover After Journal Directory Options file for database
 !  DISK:[HOME]TEST.RDB;1
 !  Created 22-JUL-2014 09:37:24.29
 !  Created by DUMP command


 TEST_J1 -
        /directory=DISK:[HOME] -
        /backup_directory=DISK:[HOME]


 TEST_J2 -
        /directory=DISK:[HOME] -
        /backup_directory=DISK:[HOME]


 TEST_J3 -
        /directory=DISK:[NEW] -
        /backup_directory=DISK:[NEW]


 TEST_J4 -
        /directory=DISK:[NEW] -
        /backup_directory=DISK:[NEW]
 !  Recover Row Cache Directory Options file for database
 !  DISK:[HOME]TEST.RDB;1
 !  Created 22-JUL-2014 09:37:24.29
 !  Created by DUMP command

 TEST_A1 -
        /directory=DISK:[HOME]
```

```
  TEST_A2 -
          /directory=DISK:[HOME]


  TEST_A3 -
          /directory=DISK:[NEW]


  TEST_A4 -
          /directory=DISK:[NEW]
%RMU-I-LOGRECDB, recovering database file DISK:[HOME]TEST.RDB;1
%RMU-I-AEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-AIJSUCCES, database recovery completed successfully
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence number
needed will be 1
%RMU-I-AIJNOENABLD, after-image journaling has not yet been enabled
```

5.1.9 RMU Unload Record_Definition File Can Include Offset and Length Comment

The record definition (.rrd) file created by the RMU Unload Record_Definition command has been enhanced to include a comment containing each field length and offset within the output record.

This optional information is included when the qualifier /DEBUG_OPTIONS=OFFSET is included on the command line. The following example shows the comment string for each field:

```
$       RMU/UNLOAD-
            /RECORD=(FILE=SALARY_HISTORY)-
            /DEBUG_OPTIONS=OFFSET-
            PERSONNEL -
            SALARY_HISTORY -
            SALARY_HISTORY
%RMU-I-DATRECUNL,   729 data records unloaded
$       type SALARY_HISTORY.RRD
DEFINE FIELD EMPLOYEE_ID DATATYPE IS TEXT SIZE IS 5.
DEFINE FIELD SALARY_AMOUNT DATATYPE IS SIGNED LONGWORD SCALE -2.
DEFINE FIELD SALARY_START DATATYPE IS DATE.
DEFINE FIELD SALARY_END DATATYPE IS DATE.
DEFINE RECORD SALARY_HISTORY.
   EMPLOYEE_ID .                        /* Offset = 0 Length = 5 */
   SALARY_AMOUNT .                      /* Offset = 5 Length = 4 */
   SALARY_START .                       /* Offset = 9 Length = 8 */
   SALARY_END .                         /* Offset = 17 Length = 8 */
END SALARY_HISTORY RECORD.              /* Total Length = 25 */
$
```

5.1.10 New RMU/DUMP/BACKUP Enhanced Error Handling Features

When the RMU/DUMP/BACKUP command detected a non−fatal error as it was reading an Oracle Rdb database backup file, it reported the error but continued with the dump to determine if there were other errors in the backup file. In addition, RMU/DUMP/BACKUP returned a success status in the $STATUS symbol when it finished reading the backup file and had a normal termination, whether or not it had output errors it detected while reading the backup file.

If the RMU/DUMP/BACKUP command is being used just to verify the validity of the backup file, reading the entire backup file just to determine if it is valid can take a long time for large backup files, especially if they are on tape media. In addition, the success status in the $STATUS symbol when the dump completed sometimes caused errors output during the dump to be missed or unnecessary time to be spent searching for any errors in an RMU/DUMP/BACKUP batch job log file.

To fix these problems, the RMU/DUMP/BACKUP error handling has been enhanced. The last most serious error detected by RMU/DUMP/BACKUP during the dump of the backup file will now always be put in the symbol $STATUS which can be tested when RMU/DUMP/BACKUP completes or aborts by executing the VMS command "SHOW SYMBOL $STATUS". In addition, a new [NO]EXIT_ERROR qualifier has been added to the RMU/DUMP/BACKUP command to optionally abort the dump operation as soon as an error is detected reading the backup file.

```
[NO]EXIT_ERROR
```

NOEXIT_ERROR, the default, keeps the current functionality: the RMU/DUMP/BACKUP operation will only be aborted if a fatal error is detected which prevents RMU/DUMP/BACKUP from continuing to dump the database backup file.

In the following example, the /EXIT_ERROR qualifier is specified with the RMU/DUMP/BACKUP command. When the first error is detected while reading the backup file, MF_PERSONNEL.RBF, the dump operation is aborted and the status of the error which caused the dump to be aborted, RMU−E−BLOCKLOST, is saved in the $STATUS symbol when the RMU/DUMP/BACKUP command exits.

```
$ RMU/DUMP/BACKUP/EXIT_ERROR MF_PERSONNEL.RBF
%RMU−I−DMPTXT_163, No dump option selected. Performing read check.
%RMU−E−BLOCKLOST, block of DEVICE:[DIRECTORY]MF_PERSONNEL.RBF; lost due to
unrecoverable error
%RMU−F−FATALERR, fatal error on DUMP_BACKUP
%RMU−F−FTL_DUMP, Fatal error for DUMP operation at  1−MAY−2013 11:32:13.45
$ SHOW SYMBOL $STATUS
  $STATUS == "%X12C8821A"
```

In the following example, the /NOEXIT_ERROR qualifier is first specified with the RMU/DUMP/BACKUP command. This is the default so the second RMU/DUMP/BACKUP command, which does not specify the /NOEXIT_ERROR qualifier, has the same results as the first RMU/DUMP/BACKUP command which does specify the /NOEXIT_ERROR qualifier. When non−fatal errors are detected reading the backup file, MF_PERSONNEL.RBF, the dump operation continues and is not aborted. But now the status of the last most severe error detected reading the backup file, RMU−E−BLOCKLOST, is saved in the $STATUS symbol when the RMU/DUMP/BACKUP command finishes reading the backup file, not a success status. A count is also given of any soft media errors which were not reported because they did not reoccur when retrying the media read operations.

```
$ RMU/DUMP/BACKUP/NOEXIT_ERROR MF_PERSONNEL.RBF
%RMU−I−DMPTXT_163, No dump option selected. Performing read check.
%RMU−E−BLOCKLOST, block of DEVICE:[DIRECTORY]MF_PERSONNEL.RBF; lost due to
unrecoverable error
%RMU−I−SOFTRERRS, 5 recoverable media errors occurred reading
DEVICE:[DIRECTORY]MF_PERSONNEL.RBF;
$ SHOW SYMBOL $STATUS
```

```
  $STATUS == "%X12C8821A"
$ RMU/DUMP/BACKUP MF_PERSONNEL.RBF
%RMU−I−DMPTXT_163, No dump option selected. Performing read check.
%RMU−E−BLOCKLOST, block of DEVICE:[DIRECTORY]MF_PERSONNEL.RBF; lost due to
unrecoverable error
%RMU−I−SOFTRERRS, 5 recoverable media errors occurred reading
DEVICE:[DIRECTORY]MF_PERSONNEL.RBF;
$ SHOW SYMBOL $STATUS
  $STATUS == "%X12C8821A"
```

5.1.11 New REVERSE Attribute for CREATE SEQUENCE Statement and IDENTITY Clause

This release of Oracle Rdb adds support for a new type of sequence. The REVERSE clause causes the value returned by NEXTVAL and CURRVAL to be bit/byte reversed. While the sequence of values computed internally by the sequence generator are regularly increasing, the values presented through the CURRVAL and NEXTVAL pseudo columns, and assigned to IDENTITY columns may not be adjacent. The advantage of such a sequence is scattered I/O when SORTED or SORTED RANKED indices are defined on such columns. This scattering of values may reduce I/O contention on nodes containing the new values generated from a normal sequence.

The new REVERSE keyword can be used in CREATE SEQUENCE, or as part of the IDENTITY clause of CREATE and ALTER TABLE statements.

The following example shows creating a table with an identity clause.

```
SQL> create table T3
cont>     (a bigint identity (reverse
cont>                         increment by 1000000
cont>                         start with −1000000)
cont>     ,rel_id integer);
SQL> insert into T3
cont>     select rel_id
cont>      from relations
cont>      order by 1 fetch
cont>      first 10 rows only;
10 rows inserted
SQL>
SQL> select t3.currval from rdb$database;

    20369552416178176
1 row selected
SQL>
SQL> table t3 order by a;
                    A        REL_ID
                    0             2
    20369552416178176            12
    40739104832356352             6
    81478209664712704             4
   122118358450569216             8
   162956419329425408             3
   203279908766482432             7
   244236716901138432             5
   269389144898142207             1
   284665759454461952             9
10 rows selected
```

```
SQL>
```

Usage Notes

- Sequences created using REVERSE generate a full 64 bit value, so columns should be created as BIGINT. Allocating a target data type that is too small will result in an *integer overflow* error as shown in the following example.

  ```
  SQL> create table T2
  cont>     (a integer identity (reverse increment by 20)
  cont>     ,rel_id integer);
  cont> insert into T2 select rel_id from relations;
  %RDB−E−ARITH_EXCEPT, truncation of a numeric value at runtime
  −COSI−F−INTOVF, integer overflow
  ```
- The REVERSE clause is incompatible with RANDOMIZE.
- REVERSE sequences are maintained as a normal sequence. RDB$NEXT_SEQUENCE_VALUE will return the current last value, but not bit/byte reversed.

Chapter 6
Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.1

6.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.1

6.1.1 New LIMIT_TO Qualifier Added to RMU Load Command

This release of Oracle Rdb adds a /LIMIT_TO qualifier to the RMU Load command.

The LIMIT_TO qualifier defines the maximum number of rows to read from the source data file. Depending upon the value specified for the SKIP qualifier, this also controls the number of rows written to the database.

The value of LIMIT_TO may not be zero, and the value of SKIP may not exceed this limit.

The default is NOLIMIT_TO which indicates that all rows read from the unload data file can be inserted into the database table, depending on other factors such as the value of the SKIP qualifier.

The following example shows loading a sample from the EMPLOYEES table using the LIMIT_TO qualifier.

```
$       RMU/LOAD -
            /LIMIT_TO=80 -
            /RECORD=(FILE:EMP_TXT,FORMAT:DELIMIT) -
            PERSONNEL_SAMPLE -
            EMPLOYEES -
            EMP.TXT
  DEFINE FIELD EMPLOYEE_ID DATATYPE IS TEXT SIZE IS 5.
  DEFINE FIELD LAST_NAME DATATYPE IS TEXT SIZE IS 14.
  DEFINE FIELD FIRST_NAME DATATYPE IS TEXT SIZE IS 10.
  DEFINE FIELD MIDDLE_INITIAL DATATYPE IS TEXT SIZE IS 1.
  DEFINE FIELD ADDRESS_DATA_1 DATATYPE IS TEXT SIZE IS 25.
  DEFINE FIELD ADDRESS_DATA_2 DATATYPE IS TEXT SIZE IS 25.
  DEFINE FIELD CITY DATATYPE IS TEXT SIZE IS 20.
  DEFINE FIELD STATE DATATYPE IS TEXT SIZE IS 2.
  DEFINE FIELD POSTAL_CODE DATATYPE IS TEXT SIZE IS 5.
  DEFINE FIELD SEX DATATYPE IS TEXT SIZE IS 1.
  DEFINE FIELD BIRTHDAY DATATYPE IS TEXT SIZE IS 16.
  DEFINE FIELD STATUS_CODE DATATYPE IS TEXT SIZE IS 1.
  DEFINE RECORD EMPLOYEES.
     EMPLOYEE_ID .
     LAST_NAME .
     FIRST_NAME .
     MIDDLE_INITIAL .
     ADDRESS_DATA_1 .
     ADDRESS_DATA_2 .
     CITY .
     STATE .
     POSTAL_CODE .
     SEX .
     BIRTHDAY .
     STATUS_CODE .
  END EMPLOYEES RECORD.
%RMU-I-DATRECREAD,  80 data records read from input file.
%RMU-I-DATRECSTO,   80 data records stored 14-OCT-2013 07:14:03.52.
$
```

This enhancement has been added in Oracle Rdb Release 7.3.1.1.

6.1.2 New BEFORE and SINCE Qualifiers Added to RMU Load Audit

Bug 17859712

This release of Oracle Rdb adds new qualifiers to RMU Load Audit to allow audit records to be filtered by timestamp. The qualifiers BEFORE and SINCE can specify the date/time range which will be extracted from the OpenVMS audit journal and saved in the target auditing table.

These qualifiers accept the standard OpenVMS date/time specification that includes special keywords such as YESTERDAY, TODAY and TOMORROW. These values can be very effective when used with the List_Plan qualifier and used later when using RMU Load Plan.

If these qualifiers are omitted, then their values default to minimum and maximum possible date/time values.

This example shows the use of DCL symbols to be used at runtime to provide the date/time range.

```
$  RMU/LOAD/AUDIT -
    /SINCE=&start_ts -
    /BEFORE=&end_ts -
   TESTDB AUDIT_RECORDS -
   SYS$COMMON:[SYSMGR]SECURITY.AUDIT$JOURNAL
%RMU-I-DATRECREAD,  91 data records read from input file.
%RMU-I-DATRECSTO,   63 data records stored 27-NOV-2013 00:59:33.18.
$
```

This example uses explicit date and time values to load a specific range of audit records.

```
$  RMU/LOAD/AUDIT -
    /SINCE=1-JAN-2011 -
    /BEFORE="1-NOV-2013 13:00" -
   TESTDB AUDIT_RECORDS -
   SYS$COMMON:[SYSMGR]SECURITY.AUDIT$JOURNAL
%RMU-I-DATRECREAD,  91 data records read from input file.
%RMU-I-DATRECSTO,   0 data records stored 27-NOV-2013 00:59:33.75.
$
```

Additionally, RMU/LOAD/PLAN now supports the AUDIT keyword and the new associated BEFORE and SINCE keywords that correspond to this new functionality.

```
$  RMU/LOAD/AUDIT -
    /SINCE=YESTERDAY -
    /NOEXECUTE -
    /LIST_PLAN=SAMPLE.PLAN -
   TESTDB AUDIT_RECORDS -
   SYS$COMMON:[SYSMGR]SECURITY.AUDIT$JOURNAL
$
```

Later the Plan can be executed and inherit the current value for YESTERDAY.

```
$  RMU/LOAD/PLAN SAMPLE.PLAN
%RMU-I-PROCPLNFIL, Processing plan file SAMPLE.PLAN.
```

```
  ! Plan created on 28-NOV-2013 by RMU/LOAD.

  Plan Name = LOAD_PLAN
  Plan Type = LOAD


  Plan Parameters:
      Database Root File = DISK1:[TESTING]TESTDB.RDB;
      Table Name = AUDIT_RECORDS
      Input File = SYS$COMMON:[SYSMGR]SECURITY.AUDIT$JOURNAL
      Audit = TESTDB
          Since = "YESTERDAY"

      ! Fields = <all>
      NoVirtual_Fields
      NoMatch_Name
      Dialect = SQL99
      Transaction_Type = PROTECTED
      ! Buffers = <default>
      ! Commit_Every = <never>
      Row_Count = 500
      ! Skip = <none>
      ! Limit_To = <none>
      NoReplace_Rows
      NoLog_Commits
      NoCorresponding
      NoDefer_Index_Updates
      Constraints
      NoParallel
      NoRestricted_Access
      NoPlace
      ! Statistics = <none>
      ! Trigger_Relations = <not specified>
  End Plan Parameters


  Executor Parameters:
      Executor Name = EXECUTOR_1
      ! Place_Only = <none>
      ! Exception_File = <none>
      ! RUJ Directory = <default>
      Communication Buffers = 1
  End Executor Parameters
%RMU-I-DATRECREAD,  195 data records read from input file.
%RMU-I-DATRECSTO,   21 data records stored 28-NOV-2013 23:34:50.27.
$
```

This enhancement has been added in Oracle Rdb Release 7.3.1.1.

6.1.3 New RMU/SHOW/STATISTICS Output File Periodic Buffer Flushes

When a system failure occurred, important diagnostic data could be lost from the Oracle Rdb RMU/SHOW STATISTICS output files: the binary file used to record Oracle Rdb database statistics for later replay; the logical area access log file; the record access dbkey log file; the process deadlock log file; the lock timeout log file; the OPCOM messages log file; the Hot Standby log file; the online analysis log file; and the stall messages log file. To minimize the loss of diagnostic data from these RMU/SHOW STATISTICS output files, periodic buffer data flushes will now occur if these files are created. These periodic output file data flushes will be the default. The user will be able to modify the periodic flush interval or specify that periodic buffer flushes are not to occur.

The syntax for this new RMU/SHOW STATISTICS qualifier is as follows.

```
/FLUSH_INTERVAL=seconds
/NOFLUSH_INTERVAL
```

The default if the new /FLUSH_INTERVAL qualifier is not specified will be a periodic flush interval of 60 seconds. The minimum flush interval that can be specified is 0 seconds. The maximum flush interval that can be specified is 3600 seconds (1 hour). Specifying 0 seconds for the flush interval is equivalent to specifying /NOFLUSH_INTERVAL. If the flush interval is active and less than the statistics collection interval, to avoid unnecessary buffer flushes the flush interval will be set to the statistics collection interval, which has a default of 3 seconds and can be set by the existing /TIME qualifier, or by typing an "S" if "Set rate" is displayed at the bottom of the statistics screen. The currrent collection interval is displayed following "Rate:" in the statistics screen header.

The first and second command examples below use the default flush interval of 60 seconds. The third and fourth command examples below specify a flush interval of 10 seconds. Note that the flush interval can be set even if the RMU/SHOW STATISTICS command does not specify any log or output files. This is because the TOOLS menu can be used later in the RMU/SHOW STATISTICS session to create log files for which the flush interval of 60 or 10 will be used.

```
$ RMU/SHOW STATISTICS MF_PERSONNEL
$ RMU/SHOW STATISTICS/OUTPUT=SHOWSTAT.DAT/DBKEY_LOG=D.LOG MF_PERSONNEL
$ RMU/SHOW STATISTICS/FLUSH_INTERVAL=10 MF_PERSONNEL
$ RMU/SHOW STATISTICS/FLUSH_INTERVAL=10/OUTPUT=SHOWSTAT.DAT/DBKEY_LOG=D.LOG
MF_PERSONNEL
```

The following command example specifies a statistics collection interval of 12 seconds using the /TIME command qualifier. This is larger than the 10 seconds specified by the new /FLUSH_INTERVAL qualifier. Since the collection interval is larger than the flush interval, the collection interval will be used for the flush interval to avoid excess buffer flushes.

```
$ RMU/SHOW STATISTICS/FLUSH_INTERVAL=10/TIME=12/OUTPUT=SHOWSTAT.DAT
/DBKEY_LOG=D.LOG MF_PERSONNEL
```

The following command examples are equivalent and specify that periodic buffer flushes will not be used for the RMU/SHOW STATISTICS binary output and log files.

```
$ RMU/SHOW STATISTICS/NOFLUSH_INTERVAL/OUTPUT=SHOWSTAT.DAT/DBKEY_LOG=D.LOG
MF_PERSONNEL
$ RMU/SHOW STATISTICS/FLUSH_INTERVAL=0/OUTPUT=SHOWSTAT.DAT/DBKEY_LOG=D.LOG
MF_PERSONNEL
```

This enhancement has been added to Oracle Rdb Release 7.3.1.1.

6.1.4 New Error and Log Messages Added for Segmented String Verification

To verify segmented strings for all Oracle Rdb database tables, the commands RMU/VERIFY/ALL or RMU/VERIFY/SEGMENTED_STRINGS/LAREAS or

RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=* can be used. To verify segmented strings for individual tables, the /LAREAS qualifier must be used with the /SEGMENTED_STRINGS qualifier to specify one or more names of tables to be verified.

There was a problem where, if the /LAREAS qualifier was used with the /SEGMENTED_STRINGS qualifier, logical area identifier numbers or the names of logical areas that were not tables could be specified with the /LAREAS qualifier without an error, even though segmented string data is verified only on a table−wide basis for a table which contains segmented string columns, including all table records that may be vertically or horizontally partitioned across multiple logical areas.

Allowing logical area id numbers or the names of logical areas that were not table names to be specified could cause confusion or lead the user to falsely assume that segmented strings contained in these logical areas were being verified. Therefore, the RMU/VERIFY operation will now be aborted and a new fatal "%RMU−I−TBLSEGVER" error will be output if the /SEGMENTED_STRINGS qualifier is specified in the same RMU/VERIFY command as the /LAREAS qualifier and the /LAREAS qualifier specifies a logical area id or a logical area name which is not a valid table name.

The following example shows the previous behavior. In the first command, "RESUMES" is a valid table that contains segmented strings and the segmented strings are therefore verified. In the second command, RMU/VERIFY detected that the "NOTATABLE" table did not exist and returned the fatal "%RMU−F−NOTLAREA" error. In the third command, the logical area id number "95" was ignored and no segmented string verification took place but the user could wrongly assume that segmented string data had been verified. In the fourth command, the index logical area name "SH_EMPLOYEE_ID" was also ignored so the user could again wrongly assume that segmented string data had been verified.

```
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=RESUMES MF_PERSONNEL
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=NOTATABLE MF_PERSONNEL
%RMU−F−NOTLAREA,  "NOTATABLE" is not a valid logical area name or number
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=95 MF_PERSONNEL
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=SH_EMPLOYEE_ID MF_PERSONNEL
```

The following example shows the new behavior. In the first command, "RESUMES" is a valid table that contains segmented strings so the segmented string data is verified as previously. In the second command, a fatal error is returned as previously but the error is detected earlier and a more specific error message is output using the new "%RMU−F−INVSEGTBL" fatal error message. In the third command and the fourth command, the invalid table names are now detected and the new "%RMU−F−INVSEGTBL" fatal error message is output.

```
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=RESUMES MF_PERSONNEL
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=NOTATABLE MF_PERSONNEL
%RMU−F−INVSEGTBL, Invalid table name NOTATABLE specified for segmented string
verification
%RMU−F−FTL_VER, Fatal error for VERIFY operation at 27−JAN−2014 15:09:05.08
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=95 MF_PERSONNEL
%RMU−F−INVSEGTBL, Invalid table name 95 specified for segmented string
verification
%RMU−F−FTL_VER, Fatal error for VERIFY operation at 27−JAN−2014 15:02:39.04
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=SH_EMPLOYEE_ID MF_PERSONNEL
%RMU−F−INVSEGTBL, Invalid table name SH_EMPLOYEE_ID specified for segmented
string verification
%RMU−F−FTL_VER, Fatal error for VERIFY operation at 27−JAN−2014 15:03:42.15
```

A new LOG message has also been added to RMU/VERIFY to list the tables containing columns defined for segmented string data for which the segmented string data will be verified. If log messages are activated for the RMU/VERIFY command, the new log message "%RMU−I−TBLSEGVER" will be output at the beginning of the verify operation and will be repeated for each table selected for segmented string data verification. In the first command, only the "RESUMES" table specified by the /LAREA qualifier is verified. In the second command, "RMU/VERIFY/ALL" is specified which, by default, verifies all tables in the database with segmented string columns, including the system tables. Note that in the second command, most of the log messages that follow the "%RMU−I−TBLSEGVER" messages have been left out to save space.

```
$ RMU/VERIFY/LOG/SEGMENTED_STRINGS/LAREAS=RESUMES MF_PERSONNEL
%RMU−I−BGNROOVER, beginning root verification
%RMU−I−ENDROOVER, completed root verification
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RESUMES
%RMU−I−DBBOUND,   bound to database "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1"
%RMU−I−OPENAREA,  opened storage area RDB$SYSTEM for protected retrieval
%RMU−I−BGNAIPVER, beginning AIP pages verification
%RMU−I−ENDAIPVER, completed AIP pages verification
%RMU−I−BGNABMSPM, beginning ABM pages verification
%RMU−I−OPENAREA,  opened storage area MF_PERS_SEGSTR for protected retrieval
%RMU−I−ENDABMSPM, completed ABM pages verification
%RMU−I−BSGPGLARE, beginning verification of RESUMES logical area
                  as part of EMP_INFO storage area
%RMU−I−OPENAREA,  opened storage area EMP_INFO for protected retrieval
%RMU−I−ESGPGLARE, completed verification of RESUMES logical area
                  as part of EMP_INFO storage area
%RMU−I−CLOSAREAS, releasing protected retrieval lock on all storage areas
%RMU−S−ENDVERIFY, elapsed time for verification :    0 00:00:00.80
$ RMU/VERIFY/ALL/LOG MF_PERSONNEL
%RMU−I−BGNROOVER, beginning root verification
%RMU−I−ENDROOVER, completed root verification
%RMU−I−TBLSEGVER, Segmented strings will be verified for table CANDIDATES
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RESUMES
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RDB$COLLATIONS
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RDB$CONSTRAINTS
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RDB$DATABASE
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RDB$FIELDS
%RMU−I−TBLSEGVER, Segmented strings will be verified for table
RDB$FIELD_VERSIONS
%RMU−I−TBLSEGVER, Segmented strings will be verified for table
RDB$INDEX_SEGMENTS
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RDB$INDICES
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RDB$MODULES
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RDB$PARAMETERS
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RDB$PRIVILEGES
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RDB$PROFILES
%RMU−I−TBLSEGVER, Segmented strings will be verified for table
RDB$QUERY_OUTLINES
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RDB$RELATIONS
%RMU−I−TBLSEGVER, Segmented strings will be verified for table
RDB$RELATION_CONSTRAINTS
%RMU−I−TBLSEGVER, Segmented strings will be verified for table
RDB$RELATION_FIELDS
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RDB$ROUTINES
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RDB$SEQUENCES
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RDB$STORAGE_MAPS
%RMU−I−TBLSEGVER, Segmented strings will be verified for table
RDB$STORAGE_MAP_AREAS
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RDB$TRIGGERS
%RMU−I−TBLSEGVER, Segmented strings will be verified for table
RDB$TRIGGER_ACTIONS
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RDB$TYPES
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RDB$TYPE_FIELDS
%RMU−I−BGNVCONST, beginning verification of constraints for database
```

```
DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
%RMU-I-ENDVCONST, completed verification of constraints for database
DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
%RMU-I-DBBOUND,   bound to database "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1"
$
```

Chapter 7
Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.0

7.1 Enhancements And Changes Provided in Oracle Rdb Release 7.3.1.0

7.1.1 Changes to Default and Limits Behavior in Oracle Rdb

This release of Oracle Rdb changes the following default behavior.

CREATE DATABASE Statement

These new defaults will be used when creating a database.

- The default PAGE SIZE changes from 2 to 4 blocks
- The default BUFFER SIZE changes from 3 pages to 4 pages
- The default NUMBER OF BUFFERS changes from 20 to 250 buffers
- The default NUMBER OF RECOVERY BUFFERS changes from 20 to 250 buffers
- The default for SECURITY CHECKING clause is now (PERSONA IS ENABLED)
- The default for SYSTEM INDEX is now (TYPE IS SORTED RANKED, COMPRESSION IS ENABLED)

The SYSTEM INDEX changes are applied to all new databases created with CREATE DATABASE statement and IMPORT DATABASE statement. Older databases converted using RMU/CONVERT or RMU/RESTORE (with the implicit Convert action) will also result in the new SYSTEM INDEX default.

The other new defaults do not affect databases created in Rdb V7.2 (or older versions) that are converted to Oracle Rdb V7.3.1 (or later) using RMU/CONVERT or RMU/RESTORE. In most cases, recreating databases using the SQL IMPORT DATABASE statement using an interchange file (.rbr) from older versions of Oracle Rdb will preserve the settings from the source database.

Interchanges files (.rbr) created with Oracle Rdb SQL EXPORT from V7.2.4 and later do export the PAGE SIZE of the database. However, older versions of Rdb did not export the PAGE SIZE if it was 2 pages (the old default). If you are using older interchange files then the IMPORT DATABASE statement should include PAGE SIZE definitions for the database and each storage area that used PAGE SIZE 2. Tools such as RMU/EXTRACT/ITEM=IMPORT can be used to create a script for this purpose.

These new limits are now enforced by Oracle Rdb.

- The maximum BUFFER SIZE can now be specified up to 256 blocks.
  Previously, the maximum allowed database buffer size was 128 blocks. Be aware that using larger database buffer sizes will require additional virtual memory.
- The minimum NUMBER OF USERS is now 5.
  In prior versions of Oracle Rdb, the minimum number of allowed database users was one (1). This minimum has been increased to five to allow for various optional database servers (such as the ABS or RCS or ALS) to access the database.
  When RMU Convert or SQL IMPORT DATABASE are used to create databases in Rdb Release 7.3, they will automatically establish a new minimum if the one defined for the original database was less than 5 users.

ALTER DATABASE Statement

When adding a new storage area to a database, that new storage area will assume a default PAGE SIZE of 4 blocks. This may be problematic if the database has a small (for instance the default) BUFFER SIZE from older database versions.

In this example, a database created under Oracle Rdb Release 7.2.5.3 was converted to Rdb Release 7.3.1.0.

```
SQL> alter database filename ABC add storage area XYZ;
SQL> attach 'filename ABC';
SQL> show storage area XYZ

     XYZ
         Access is:      Read write
         Page Format:    Uniform
         Page Size:      4 blocks
         Area File:      USER2:[TESTING]XYZ.RDA;1
         Area Allocation:          700 pages
         Extent:         Enabled
         Area Extent Minimum:      99 pages
         Area Extent Maximum:      9999 pages
         Area Extent Percent:      20 percent
         Snapshot File:  USER2:[TESTING]XYZ.SNP;1
         Snapshot Allocation:      100 pages
         Snapshot Extent Minimum:  99 pages
         Snapshot Extent Maximum:  9999 pages
         Snapshot Extent Percent:  20 percent
         Locking is Row Level
         No Cache Associated with Storage Area
No database objects use Storage Area XYZ
SQL>
```

The problem lies in the fact that the current BUFFER SIZE is only 6 blocks (see the SHOW DATABASE output below). This would mean that I/O to the new storage area would only be adding one page to the buffer, with over 33% of the buffer wasted.

```
SQL> show database rdb$dbhandle
Default alias:
    Oracle Rdb database in file ABC
        Multischema mode is disabled
        Number of users:              50
        Number of nodes:              16
        Buffer Size (blocks/buffer):   6
        Number of Buffers:            20
        Number of Recovery Buffers:   20
   .
   .
   .
```

Oracle recommends that an explicit PAGE SIZE clause be used when defining a new storage area.

CREATE INDEX Statement

These new defaults will be used when creating an index.

- The default NODE SIZE for SORTED RANKED and unique SORTED indices is now chosen to be large enough to fill the free space in the new logical area. In prior versions, a smaller NODE SIZE was chosen based on the index key length. However, these small nodes left unused space on the database pages and so were wasteful of disk space and virtual memory (when in buffers).

No changes were made to the default node size for SORTED indices with duplicates. In this case, the smaller duplicate nodes may fill the unused space on the page.
- The default PERCENT FILL changes from 70% to 85%

These changes do not affect indices created in Rdb V7.2 (or older versions) when the database is converted to Oracle Rdb Release 7.3.1 using RMU/CONVERT or RMU/RESTORE.

Importing a database using the IMPORT DATABASE statement will fix up the metadata for the PERCENT FILL and NODE SIZE so that the output from SHOW INDEX will provide more information than in prior versions.

Logical Name RDMS$BIND_WORK_VM

The Oracle Rdb query optimizer might make use of temporary virtual memory (VM) during query processing. This memory is used to cache index keys during zig−zag match strategy and dbkey lists during temporary−relation processing (not related to the SQL temporary table feature). In either case, the virtual memory size defined by the logical name RDMS$BIND_WORK_VM is used for each buffer which might overflow to a temporary disk file (located using the RDMS$BIND_WORK_FILE logical name).

With this release the new default has increased from 10,000 to 100,000 bytes. This change makes it more likely that queries can complete entirely in VM rather than opening and using a small disk file.

Logical Name RDMS$BIND_MAX_QSORT_COUNT

When the number of rows is relatively small, the Oracle Rdb query processor can avoid using SORT32 (which has a higher setup cost) by using an in−memory Quick Sort. In prior versions, the default threshold was 63 rows. This release of Oracle Rdb defaults to 5,000 rows. The larger threshold should allow more sorting to consume less resources. This threshold can be changed (for instance to return to the prior default of 63) using the logical name RDMS$BIND_MAX_QSORT_COUNT.

7.1.2 New /ERROR_LIMIT Qualifier Added as the Default to RMU/VERIFY

New default functionality has been added to RMU/VERIFY to limit the number of diagnostics output when verifying an Oracle Rdb database. By default, RMU/VERIFY will now limit the number of diagnostic messages output by RMU/VERIFY to 100. If this limit is exceeded, the RMU/VERIFY will terminate with a warning messsage.

```
%RMU−W−MAXVERERR, Maximum error limit 100 exceeded − ending verification
```

To disable this behavior and have no limit on the number of diagnostic messages output by the verification, /NOERROR_LIMIT must be specified on the command line.

```
RMU/VERIFY/ALL/NOERROR_LIMIT mf_personnel
```

To override the default of 100, the user can specify a numeric value for the /ERROR_LIMIT between 1 and 2147483647.

```
RMU/VERIFY/ALL/ERROR_LIMIT=50 mf_personnel
```

If /ERROR_LIMIT is specified without a value, the default limit of 100 diagnostics will be used.

```
RMU/VERIFY/ALL/ERROR_LIMIT mf_personnel
```

The /ERROR_LIMIT does not include any logging messages put out when the /LOG qualifier is used with RMU/VERIFY. It only includes diagnostic messages which are defined as messages of any severity which are not logging messages put out when the /LOG qualifier is specified. The %RMU−W−MAXVERERR message is not included in the /ERROR_LIMIT count but is output once the /ERROR_LIMIT in force is exceeded in place of the diagnostic message that would have exceeded the error limit.

We have done everything we can to make the /ERROR_LIMIT count as accurate as possible but related messages output together in one output operation may be counted as one message in a limited number of cases. Therefore, we do not guarantee absolute accuracy in the /ERROR_LIMIT count in all cases but consider it as an acceptably accurate way to limit the potentially large number of diagnostics that can be output by the RMU/VERIFY of a database.

The syntax for this qualifier is as follows:

```
/[NO]ERROR_LIMIT[=n]
```

"n" is a positive numeric value between 1 and 2147483647.

In the following example, the RMU/VERIFY of a database completes normally with 6 diagnostics since the default error limit of 100 is not exceeded.

```
$ rmu/verify/all mf_personnel
%RMU−W−PAGCKSBAD, area EMP_INFO, page 2
                  contains an invalid checksum
                  expected: A77B3D6D, found: A7713D6D
%RMU−W−PAGLIXFRS, area EMP_INFO, page 2
                  line index entry 15 maps free space at offset 00000106 (hex)
%RMU−W−PAGLIXSML, area EMP_INFO, page 2
                  line index entry 19, length too small
                  expected at least 2, found: 0
%RMU−W−PAGLIXSML, area EMP_INFO, page 2
                  line index entry 20, length too small
                  expected at least 2, found: 0
%RMU−W−PAGLIXSML, area EMP_INFO, page 2
                  line index entry 21, length too small
                  expected at least 2, found: 0
%RMU−W−PAGLIXSML, area EMP_INFO, page 2
                  line index entry 22, length too small
                  expected at least 2, found: 0
$
```

In the following example, the RMU/VERIFY is of the same database as in the previous example but an error limit of 5 diagnostic messages is specified. Therefore, 5 diagnostic messages are output and instead of the 6th diagnostic message being output, the %RMU−W−MAXVERERR is output and the database verify terminates.

```
$ rmu/verify/all/error_limit=5 mf_personnel
%RMU−W−PAGCKSBAD, area EMP_INFO, page 2
                  contains an invalid checksum
                  expected: A77B3D6D, found: A7713D6D
%RMU−W−PAGLIXFRS, area EMP_INFO, page 2
                  line index entry 15 maps free space at offset 00000106 (hex)
```

```
%RMU−W−PAGLIXSML, area EMP_INFO, page 2
                  line index entry 19, length too small
                  expected at least 2, found: 0
%RMU−W−PAGLIXSML, area EMP_INFO, page 2
                  line index entry 20, length too small
                  expected at least 2, found: 0
%RMU−W−PAGLIXSML, area EMP_INFO, page 2
                  line index entry 21, length too small
                  expected at least 2, found: 0
%RMU−W−MAXVERERR, Maxium error limit 5 exceeded − ending verification
```

7.1.3 RMU /VERIFY Root Displays the Corrupt Page Table Entries

Bug 870984

In previous releases of Oracle Rdb, the command RMU/VERIFY ROOT did not show any Corrupt Page Table (CPT) entries even when they existed. The commands RMU/SHOW CORRUPT, RMU/DUMP/HEAD=CORRUPT and RMU/VERIFY/ALL would show them.

A new feature has been added to RMU/VERIFY ROOT so that it now displays the CPT entries.

The following example shows that RMU/VERIFY/ROOT now displays the corrupt page entries in the database corrupt page table. The RMU/SHOW CORRUPT command is first executed to display the corrupt page entries in the corrupt page table for the PERSONNEL database. When the RMU/VERIFY/ROOT command is then executed it outputs a message for each corrupt page entry in the corrupt page table. This is the same message output by the RMU/VERIFY/ALL command.

```
$ RMU/SHOW CORRUPT PERSONNEL
*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
* Oracle Rdb V7.3−100                                     1−FEB−2013 16:41:40.77
*
* Dump of Corrupt Page Table
*     Database: DEVICE:[DIRECTORY]PERSONNEL.RDB;2
*
*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Entries for storage area RDB$SYSTEM
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

    Page 300
        − AIJ recovery sequence number is −1
        − Live area ID number is 1
        − Consistency transaction sequence number is 0
        − State of page is: corrupt


*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
* Oracle Rdb V7.3−100                                     1−FEB−2013 16:41:40.77
*
* Dump of Storage Area State Information
*     Database: DEVICE:[DIRECTORY]PERSONNEL.RDB;2
*
*−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

All storage areas are consistent.
```

```
$ RMU/VERIFY/ROOT/LOG PERSONNEL
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-DBBOUND,   bound to database "DEVICE:[DIRECTORY]PERSONNEL.RDB;2"
%RMU-I-OPENAREA,  opened storage area DEVICE:[DIRECTORY]PERSONNEL.RDB;2 for protected retrieval
%RMU-I-BGNAIPVER, beginning AIP pages verification
%RMU-I-ENDAIPVER, completed AIP pages verification
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-I-ENDABMSPM, completed ABM pages verification
%RMU-E-CORRUPTPG, Page 300 in area DEVICE:[DIRECTORY]PERSONNEL.RDB;2 is marked as corrupt.
%RMU-I-CLOSAREAS, releasing protected retrieval lock on all storage areas
%RMU-S-ENDVERIFY, elapsed time for verification :    0 00:00:00.49
$
```

7.1.4 DECLARE LOCAL TEMPORARY TABLE Supports COMMENT IS Clause

With this release of Oracle Rdb, the DECLARE LOCAL TEMPORARY TABLE statement now supports the COMMENT IS clause. This comment is not stored by Rdb but can be used to document the DECLARE statement when it appears in a CREATE MODULE statement or when used in Interactive SQL scripts.

The following example shows the placement of the clause.

```
SQL> declare local temporary table module.STBL
cont>     (a int)
cont>     on commit preserve rows
cont>     comment is 'Test for local temporary table'
cont>     large memory is enabled
cont> ;
```

For further details please refer to the Oracle Rdb SQL Reference Manual.

7.1.5 Temporary Tables Now Support LARGE MEMORY Option

With this release of Oracle Rdb, the DECLARE LOCAL TEMPORARY TABLE statement, the CREATE GLOBAL TEMPORARY TABLE statement, and the CREATE LOCAL TEMPORARY TABLE statement all support the use of LARGE MEMORY on OpenVMS.

A new LARGE MEMORY IS { ENABLED | DISABLED } clause has been added to these statements so that the temporary table virtual memory now resides in 64 bit memory. This allows much larger temporary tables than in previous releases of Oracle Rdb.

The following example shows the placement of the clause.

```
SQL> create local temporary table LTBL
cont>     (a int)
cont>     on commit preserve rows
cont>     comment is 'Test for local temporary table'
cont>     large memory is enabled
cont> ;
SQL> show table (column) LTBL;
```

```
 Information for table LTBL

A local temporary table.
On commit Preserve rows
        Large Memory:          Enabled

Columns for table LTBL:
Column Name                     Data Type        Domain
-----------                     ---------        ------
A                               INTEGER

SQL>
```

Additionally, the ALTER TABLE statement has been enhanced to enable (or disable) this feature on existing temporary table definitions. This clause is not permitted for information on base tables.

For further details please refer to the Oracle SQL Reference Manual.

7.1.6 COUNT Now Returns BIGINT Result

The SQL aggregate function COUNT now returns BIGINT (aka 64 bit values) in this release of Oracle Rdb (Release 7.3.1.0). This change has been made to accommodate large tables and result sets.

The side effects of this change that may be visible are:

- Wider column output in interactive SQL for queries that perform SELECT COUNT(*), COUNT(DISTINCT expression) and COUNT(expression).
- COMPUTED BY and AUTOMATIC columns will now be displayed as BIGINT type because of an expression that uses a COUNT function.
- SQLDA data type change for COUNT expressions.

In general, this change is backward compatible with existing applications. The computed BIGINT value will automatically be converted to INTEGER for older SQL precompiler or SQL module language applications.

7.1.7 Aggregate Functions Now Use BIGINT Counters

COUNT and related aggregate functions AVG, VARIANCE, and STDDEV all process counters in BIGINT registers. This allows Rdb to process aggregation across much larger row sets than in previous releases.

With this release of Oracle Rdb, the COUNT aggregate function will return BIGINT data type results. In prior releases, an INTEGER type was the result. Applications that create COMPUTED BY or AUTOMATIC columns may notice that the data type of such columns changed to BIGINT.

Note

Oracle Rdb will implicitly convert internal results to INTEGER if the target data type in the application has not changed.

7.1.8 /[NO]KEY_VALUES Qualifier Added to RMU/VERIFY/INDEX

A new /KEY_VALUES qualifier has been added to the Oracle Rdb RMU/VERIFY command for verifying the integrity of Rdb databases. The /KEY_VALUES qualifier verifies the key field values contained in a sorted, sorted ranked or hashed index against the key field values in the matching table row to make sure the key field values contained in the index match the field values in the table. This qualifier can only be specified if indexes are being verified explicitly as with the RMU/VERIFY/INDEX command or if indexes are being verified by default such as with the RMU/VERIFY/ALL command. Key field values will be verified for all indexes or a specified list of one or more indexes. Diagnostic error messages will be put out if the index key field value does not match the table row key field value, if a DBKEY contained in the index node or hash bucket does not match the DBKEY of a table row, or if the DBKEY of a table row is not contained in the index node or hash bucket. Indexes with one key field or multiple key fields can be verified using the /KEY_VALUES qualifier.

The /KEY_VALUES qualifier will ignore index fields that have a collating sequence or are of type character varying. The character encoding for the collating sequence prevents a byte by byte comparison with the column row values. The key encoding for character varying pads with spaces so the precise length cannot be compared with the column row values. For these index fields, a warning message will be output and a comparison with the column row values will not be done.

The syntax for this new qualifier is as follows −

```
/[NO]KEY_VALUES
```

Note that /KEY_VALUES is not the default and must be specified.

The following example shows the error message put out if the index key field value in the index structure does not match the key field value in the table row. The DBKEY pointer to the index node or hash bucket that contains the row DBKEY is output as well as the table row DBKEY.

```
$ RMU/VERIFY/INDEX=JH_EMPLOYEE_ID/KEY_VALUES MF_PERSONNEL
%RMU−E−BADIDXFLD, Index JH_EMPLOYEE_ID key field EMPLOYEE_ID value at dbkey
93:810:3 does not match stored value for table JOB_HISTORY at dbkey 90:289:5 .
```

The following example shows the error message put out if the key field value(s) returned from index only retrieval of the key fields do not match on row DBKEY with the key fields returned sequentially from the table row. Because the retrieval is out of sequence, the key field values cannot be compared.

```
$ RMU/VERIFY/INDEX/KEY_VALUES MF_PERSONNEL
%RMU−E−BADIDXDBK, Index JH_EMPLOYEE_ID dbkey 91:413:3 does not match
table JOB_HISTORY row dbkey 90:200:4 .
```

The following example shows the error message put out if the table contains a DBKEY pointing to a table row that is not in the index.

```
$ RMU/VERIFY/INDEX/KEY_VALUES/ERROR_LIMIT=200 MF_PERSONNEL
%RMU−E−NOTIDXDBK, Table COLLEGES row dbkey 68:2:1 is not in index
 COLL_COLLEGE_CODE .
```

The following example shows the error message put out if the index contains a DBKEY pointing to a table row that is not in the table.

```
$ RMU/VERIFY/INDEX/KEY_VALUES/ERROR=200 MF_PERSONNEL
%RMU−E−NOTTABDBK, Index COLL_COLLEGE_CODE dbkey 68:2:1 is not in table
 COLLEGES .
```

The following example shows that a warning message is put out for index key fields that have a collating sequence or that are of type character varying. These index fields cannot be compared with row values using the /KEY_VALUES qualifier.

```
$ RMU/VERIFY/INDEX/KEY_VALUES ABC.RDB
%RMU−W−NOTTYPNDX, Index MANUFACTURING_INDEX field MANUFACTURER_NAME cannot
 be verified with the row value in table MANUFACTURING;
 reason − COLLATING SEQUENCE.
$ RMU/VERIFY/INDEX/KEY_VALUES DBASE_INDEX.RDB
%RMU−W−NOTTYPNDX, Index FORECAST_HASH field PART_NO cannot be verified with
 the row value in table FORECAST_VOLUME; reason − CHARACTER VARYING.
```

7.1.9 The /LOCK_TIMEOUT Qualifier Now Allows the Database Default

For the Oracle Rdb RMU commands RMU/BACKUP/ONLINE, RMU/COPY/ONLINE, and RMU/BACKUP/AFTER/QUIET_POINT, the /LOCK_TIMEOUT qualifier can be specified. Previously the /LOCK_TIMEOUT qualifier required a value, the maximum time in seconds to wait for acquiring the database QUIET POINT and other locks used for online database access. If "/LOCK_TIMEOUT = n" was not specified, RMU would wait indefinitely to acquire the database lock it needed.

Now the /LOCK_TIMEOUT qualifier can be specified without a value. In this case, the default lock timeout value specified for the database will be used. Specifically, the default lock timeout value used will be the value of the logical name RDM$BIND_LOCK_TIMEOUT_INTERVAL if it has been specified, otherwise the "LOCK TIMEOUT INTERVAL" specified by the SQL CREATE DATABASE or ALTER DATABASE command will be used. If neither value has been specified, the value used will be the maximum possible lock timeout value which can be specified for an Oracle Rdb database.

The new syntax for this qualifier is as follows −

```
/LOCK_TIMEOUT [ = n ]
```

Note that /LOCK_TIMEOUT is not the default and must be specified. The default, if /LOCK_TIMEOUT is not specified, continues to be to wait indefinitely to acquire the QUIET POINT or other database locks requested by RMU.

The following example shows the different RMU commands which accept the /LOCK_TIMEOUT qualifier. In the first command, the specified lock timeout value of 100 seconds will be used. In the other commands, since a lock timeout value is not specified, the default database lock timeout value described above will now be used.

```
$ RMU/BACKUP/ONLINE/LOCK_TIMEOUT=100/NOLOG MF_PERSONNEL MFP.RBF
$ RMU/BACKUP/ONLINE/LOCK_TIMEOUT/NOLOG MF_PERSONNEL MFP.RBF
$ RMU/COPY/ONLINE/LOCK_TIMEOUT/ROOT=DISK:[DIRECTORY]MF_PERSONNEL−
  /NOAFTER/NOLOG MF_PERSONNEL
$ RMU/BACKUP/AFTER/NOLOG/QUIET_POINT/LOCK_TIMEOUT −
```

```
  DISK:[DIRECTORY]MF_PERSONNEL MFP_AIJ_1
```

The following example shows that the PLAN file used with the RMU PARALLEL BACKUP command will accept either the specified lock timeout value of 100 seconds or will now accept the default database lock timeout value described above.

```
$ RMU/BACKUP/PARALLEL=EXEC=1/DISK=WRITER=2/LIST=TMP.PLAN/ONLINE-
  /LOCK_TIMEOUT=100 MF_PERSONNEL DISK:[DIRECTORY]MFP,DISK:[DIRECTORY]
$ SEAR TMP.PLAN LOCK_TIMEOUT
    Lock_Timeout = 100
$ RMU/BACKUP/PARALLEL=EXEC=1/DISK=WRITER=2/LIST=TMP.PLAN/ONLINE-
  /LOCK_TIMEOUT MF_PERSONNEL DISK:[DIRECTORY]MFP,DISK:[DIRECTORY]
$ SEAR TMP.PLAN LOCK_TIMEOUT
    Lock_Timeout
$ RMU/BACKUP/PLAN TMP.PLAN
```

7.1.10 Compression of AIJ Backup Files for Automatic AIJ Backups

A new feature has been added to allow compression of AIJ files during automatic AIJ backups.

Also backups of after image journals using /Format=Old_File can be compressed using the ZLIB compression method.

To set the compression level, the RMU SET command has been extended:

```
$ RMU /SET AFTER_JOURNAL /BACKUPS=(...,[[NO]COMPRESSION[=ZLIB[=n]]])
  default is NOCOMPRESSION
  n = ZLIB compression level,
      default is 6, minimum is 1, maximum is 9
```

The RMU recover command has been enhanced to automatically decompress AIJ backup files in the 'Old_file' format which have been compressed using the ZLIB compression method.

7.1.11 Global Statistics Sections for Better Performance

On systems with many CPUs, updating database statistics from many application processes causes memory cache invalidation and therefore prolongs the update of the statistics data.

With the change in this release of Oracle Rdb, the RDM Monitor creates 16 global statistic sections for systems with 16 or more CPUs. Application processes attach to a statistics section based on the modulo 16 of their process ID value. This should reduce the coincidence that two or more processes use the same global section from different processors and thus cause memory cache invalidation when updating statistics data.

The default used for RAD (Resource Allocation Domain) systems still remains (see below).

The use of multiple global statistic sections can be overridden with the following system logical name:

```
$ DEFINE /SYSTEM RDM$BIND_MONITOR_GLOBAL_STATS_SECTIONS n
  n = 0 − always use statistics in the database's shared memory section
  n = 1..16 − use statistics in separate global sections
```

```
            with n the number of global sections being used
  If n is equal −1 or if the logical is not defined, use the default (see
  below).
```

By default, the statistics area in the database's shared memory section is used unless a system has more than one RAD with memory or the system has 16 or more CPUs. In the case of more than one RAD with memory, one global statistics section is created per RAD with memory. In the case of 16 or more CPUs, 16 global statistics sections are created. The more than one RAD with memory case has precedence over the 16 or more CPUs case.

7.1.12 RMU/SET AUDIT Supports Wildcard Table and Column Names

Bug 5865199

In prior versions of Oracle Rdb, the RMU Set Audit command did not allow wildcards to be used to specify tables or column names for the DACCESS qualifier. A database administrator was required to enumerate all tables and columns to be audited. Further, if a table was specified as * then this was ignored by RMU.

With this release, Rdb RMU Set Audit has been enhanced to provide more flexible naming conventions for tables and columns.

- TABLE=* and COLUMN=*.* are now accepted to specify all tables or all columns of all tables.
- Table and column names may also contain OpenVMS style wildcards "*" and "%". For instance, in the PERSONNEL database, JOB* will match both the JOBS and JOB_HISTORY table and *HISTORY.EMPLOYEE_ID will match all EMPLOYEE_ID columns in any table ending in HISTORY which includes both the JOB_HISTORY and SALARY_HISTORY tables.

The following example shows the simplified commands for enabling auditing for important fields in the PERSONNEL database.

```
$ rmu/set audit-
    /enable=daccess=column=(-
        *.employee_id,-
        *_history.*end,-
        *_history.*start)-
    /privileges=(insert,select,update,delete) personnel
$
$ rmu/show audit/daccess=(DATABASE,TABLE,COLUMN) personnel
Security auditing STOPPED for:
    DACCESS (disabled)
        DATABASE
            (NONE)
        COLUMN : DEGREES.EMPLOYEE_ID
            (SELECT,INSERT,UPDATE,DELETE)
        COLUMN : EMPLOYEES.EMPLOYEE_ID
            (SELECT,INSERT,UPDATE,DELETE)
        COLUMN : JOB_HISTORY.EMPLOYEE_ID
            (SELECT,INSERT,UPDATE,DELETE)
        COLUMN : JOB_HISTORY.JOB_START
            (SELECT,INSERT,UPDATE,DELETE)
        COLUMN : JOB_HISTORY.JOB_END
            (SELECT,INSERT,UPDATE,DELETE)
        COLUMN : RESUMES.EMPLOYEE_ID
            (SELECT,INSERT,UPDATE,DELETE)
        COLUMN : SALARY_HISTORY.EMPLOYEE_ID
            (SELECT,INSERT,UPDATE,DELETE)
        COLUMN : SALARY_HISTORY.SALARY_START
            (SELECT,INSERT,UPDATE,DELETE)
        COLUMN : SALARY_HISTORY.SALARY_END
            (SELECT,INSERT,UPDATE,DELETE)

Security alarms STOPPED for:
    DACCESS (disabled)
        DATABASE
            (NONE)
        COLUMN : DEGREES.EMPLOYEE_ID
            (SELECT,INSERT,UPDATE,DELETE)
        COLUMN : EMPLOYEES.EMPLOYEE_ID
            (SELECT,INSERT,UPDATE,DELETE)
        COLUMN : JOB_HISTORY.EMPLOYEE_ID
            (SELECT,INSERT,UPDATE,DELETE)
        COLUMN : JOB_HISTORY.JOB_START
            (SELECT,INSERT,UPDATE,DELETE)
        COLUMN : JOB_HISTORY.JOB_END
            (SELECT,INSERT,UPDATE,DELETE)
        COLUMN : RESUMES.EMPLOYEE_ID
            (SELECT,INSERT,UPDATE,DELETE)
        COLUMN : SALARY_HISTORY.EMPLOYEE_ID
            (SELECT,INSERT,UPDATE,DELETE)
        COLUMN : SALARY_HISTORY.SALARY_START
            (SELECT,INSERT,UPDATE,DELETE)
        COLUMN : SALARY_HISTORY.SALARY_END
            (SELECT,INSERT,UPDATE,DELETE)
```

Usage Notes

When using wildcard characters with /ENABLE=DACCESS or /DISABLE=DACCESS option, only user defined objects will be selected.

- TABLE − only user defined tables will be matched by the wildcard search. Views, system tables and special tables created by OCI Services for Rdb will not be returned. For excluded tables, you must specify their full name on the RMU command line.
- SEQUENCE − only user defined sequences will be matched by the wildcard search. This includes sequences implicity created for IDENTITY columns. For excluded sequences, you must specify their full name on the RMU command line.
- MODULE − only user defined modules will be matched by the wildcard search. For excluded modules, you must specify their full name on the RMU command line.
- ROUTINE − only user defined routines will be matched by the wildcard search. Any routine defined in a module with USAGE IS LOCAL will be excluded from this matching. Such routines can only be called from within the module itself and so additional auditing is not required. For excluded routines, you must specify their full name on the RMU command line.
- VIEW − only user defined views will be matched by the wildcard search. Base tables, system views, and special views created by OCI Services for Rdb will not be returned. For excluded views, you must specify their full name on the RMU command line.

7.1.13 RMU/BACKUP Database Root Verification Performance Enhancement

By default, RMU/BACKUP verifies the database root at the start of the backup before backing up an Oracle Rdb database. If the database root is invalid, the error diagnostics from the verification are output and the backup is terminated. This is to prevent backing up a corrupt database. To not verify the database root, the user must specify /NODATABASE_VERIFICATION. As part of this verification, all live and snapshot database storage areas were opened to verify the area prologue block and the area maximum page number and then closed. Later, the storage areas were again opened and closed a second time to do the actual backup of the data in each storage area.

To improve the performance of RMU/BACKUP, the extra open and close of the live and snapshot database storage areas just to verify the database root has been eliminated. The same verification is still done but the storage areas are now only opened and closed once during the backup. Note that /DATABASE_VERIFICATION remains the default for RMU/BACKUP.

As part of this change, the following new syntax has been added to the RMU/BACKUP/PARALLEL *.PLAN file.

```
[No]Database_Verification
```

The existing command line qualifier "/[NO]DATABASE_VERIFICATION" is used to set the new "[No]Database_Verification" option in the initial *.PLAN file created from the RMU/BACKUP command line. The *.PLAN file can then be edited to change this option if desired. The default is "/DATABASE_VERIFICATION".

The following example shows some of the verification diagnostics that can be output when the database root is verified by RMU/BACKUP.

```
$ RMU/BACKUP/NOLOG MF_PERSONNEL MFP.RBF
%RMU−E−BADMAXPNO, unable to read last page (1052) in file
DEVICE:[DIRECTORY]DEPARTMENTS.RDA;1
%RMU−F−ABORTVER,  fatal error encountered; aborting verification
%RMU−F−FATALERR, fatal error on BACKUP
%RMU−F−FTL_BCK, Fatal error for BACKUP operation at 11−MAY−2009
11:16:59.46
$ RMU/BACKUP/NOQUIET/ONLINE/NOLOG MF_PERSONNEL MFP
%RMU−W−BADPROID, DEVICE:[DIRECTORY]EMPIDS_MID.SNP;1
 file contains a bad identifier
 Expected "RDMSDATA", found "NOTRDBDB"
%RMU−W−INVALFILE, inconsistent database file
 DEVICE:[DIRECTORY]EMPIDS_MID.SNP;1
%RMU−F−ABORTVER,  fatal error encountered; aborting verification
%RMU−F−FATALERR, fatal error on BACKUP
%RMU−F−FTL_BCK, Fatal error for BACKUP operation at 11−MAY−2009
11:11:25.68
```

The following example shows that the RMU/BACKUP *.PLAN file "[No]Database_Verification" option is set based on the "/DATABASE_VERIFICATION" qualifier on the RMU/BACKUP command line. The default is "Database_Verification" so unless "/NODATABASE_VERIFICATION" is specified, the "Database_Verification" option will be set in the *.PLAN file.

```
$ RMU/BACKUP/PARALLEL=EXEC=1/DISK=WRITER=2/LIST=TMP.PLAN/ONLINE−
  MF_PERSONNEL DISK:[DIRECTORY1]MFP,DISK:[DIRECTORY2]
$ SEAR TMP.PLAN DATABASE_VERIFICATION
    Database_Verification
$ RMU/BACKUP/PARALLEL=EXEC=1/DISK=WRITER=2/LIST=TMP.PLAN/ONLINE−
  /DATABASE_VERIFICATION MF_PERSONNEL DISK:[DIRECTORY1]MFP,−
  DISK:[DIRECTORY2]
$ SEAR TMP.PLAN DATABASE_VERIFICATION
    Database_Verification
$ RMU/BACKUP/PARALLEL=EXEC=1/DISK=WRITER=2/LIST=TMP.PLAN/ONLINE-
  /NODATABASE_VERIFICATION MF_PERSONNEL -
  DISK:[DIRECTORY1]MFP,DISK:[DIRECTORY2]
$ SEAR TMP.PLAN DATABASE_VERIFICATION
    NoDatabase_Verification
$ RMU/BACKUP/PLAN TMP.PLAN
```

7.1.14 RMU /UNLOAD /AFTER_JOURNAL New Qualifier /DELETES_FIRST

Bug 3388774

The qualifier "/DELETES_FIRST" has been added to the RMU /UNLOAD /AFTER_JOURNAL command. Specifying "/DELETES_FIRST" indicates that all delete operations within each transaction are to be returned before any add/modify operations.

Record Order Unpredictable

Within the output stream for a transaction, the order of records returned from the LogMiner remains unpredictable.

7.1.15 Add Option to Pass Values to /CONFIRM During RESTORE Operation

Bug 411144

In prior releases of Oracle Rdb, if problems occurred during tape operation of an RMU/RESTORE command and the /CONFIRM option was selected, then the operation would wait for user input on the terminal before continuing.

```
$RMU/RESTORE/NOCDD/REWIND LMA1001:VOL002.RBF MF_PERSONNEL /DIRECTORY=[] /LOG
%RMU-I-WRNGLBL, Tape on LMA1001 was incorrectly labeled. Expected VOL002 -
Found MF_PER
%RMU-I-TAPEDISPR, Specify tape disposition for LMA1001 (QUIT,OVERRIDE,RETRY,
UNLOAD)
RMU> QUIT (User has to enter the RESPONSE.)
%RMU-F-ABORT, operator requested abort on fatal error
%RMU-F-FATALERR, fatal error on RESTORE
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 13-JUL-2013 11:22:32.90

$RMU/RESTORE/NOCDD/REWIND LMA1001:VOL002.RBF MF_PERSONNEL /DIRECTORY=[] /LOG
%RMU-I-WRNGLBL, Tape on LMA1001 was incorrectly labeled. Expected VOL002 -
Found MF_PER
%RMU-I-TAPEDISPR, Specify tape disposition for LMA1001 (QUIT,OVERRIDE,RETRY,
UNLOAD)
RMU> RETRY (User has to enter the RESPONSE.)
%MOUNT-I-MOUNTED, MF_PER mounted on LMA1001:
%RMU-I-WRNGLBL, Tape on LMA1001 was incorrectly labeled. Expected VOL002 -
```

```
Found MF_PER
%RMU−I−TAPEDISPR, Specify tape disposition for LMA1001 (QUIT,OVERRIDE,RETRY,
UNLOAD)
RMU> QUIT (User has to enter the RESPONSE.)
%RMU−F−ABORT, operator requested abort on fatal error
%RMU−F−FATALERR, fatal error on RESTORE
%RMU−F−FTL_RSTR, Fatal error for RESTORE operation at 13−JUL−2013 11:22:55.86

$RMU/RESTORE/NOCDD/REWIND LMA1001:VOL002.RBF MF_PERSONNEL /DIRECTORY=[] /LOG
%RMU−I−WRNGLBL, Tape on LMA1001 was incorrectly labeled. Expected VOL002 −
Found MF_PER
%RMU−I−TAPEDISPR, Specify tape disposition for LMA1001 (QUIT,OVERRIDE,RETRY,
UNLOAD)
RMU> OVERRIDE (User has to enter the RESPONSE.)
%RMU−F−FILACCERR, error opening input file LMA1001:[000000]VOL002.RBF;
−SYSTEM−W−NOSUCHFILE, no such file
%RMU−F−FATALERR, fatal error on RESTORE
%RMU−F−FTL_RSTR, Fatal error for RESTORE operation at 13−JUL−2013 11:23:05.59
```

A new feature has been added in this release to correct this problem. The user has the option of selecting values for /CONFIRM during a RESTORE from tape operation. The new syntax and valid values are:

```
RMU/RESTORE...   /CONFIRM[=QUIT|RETRY=x|OVERRIDE|UNLOAD]
```

See the following examples of this new feature.

```
$RMU/RESTORE/NOCDD/REWIND LMA1001:VOL002.RBF MF_PERSONNEL /DIRECTORY=[]
/LOG /CONFIRM=QUIT
%MOUNT−I−MOUNTED, MF_PER mounted on LMA1001:
%RMU−I−TAPEDEF, Terminating restore operation as requested by user
%RMU−F−ABORT, operator requested abort on fatal error
%RMU−F−FATALERR, fatal error on RESTORE
%RMU−F−FTL_RSTR, Fatal error for RESTORE operation at 13−JUL−2013 11:43:31.35

$RMU/RESTORE/NOCDD/REWIND LMA1001:VOL002.RBF MF_PERSONNEL /DIRECTORY=[]
/LOG /CONFIRM=RETRY=2
%RMU−I−TAPEDEF, Retrying tape operation as requested by user
%MOUNT−I−MOUNTED, MF_PER mounted on LMA1001:
%RMU−I−TAPEDEF, Retrying tape operation as requested by user
%MOUNT−I−MOUNTED, MF_PER mounted on LMA1001:
%RMU−F−ABORT, operator requested abort on fatal error
%RMU−F−FATALERR, fatal error on RESTORE
%RMU−F−FTL_RSTR, Fatal error for RESTORE operation at 13−JUL−2013 11:43:42.97

$RMU/RESTORE/NOCDD/REWIND LMA1001:VOL002.RBF MF_PERSONNEL /DIRECTORY=[]
/LOG /CONFIRM=OVERRIDE
%RMU−I−TAPEDEF, Overriding tape label as requested by user
%RMU−F−FILACCERR, error opening input file LMA1001:[000000]VOL002.RBF;
−SYSTEM−W−NOSUCHFILE, no such file
%RMU−F−FATALERR, fatal error on RESTORE
%RMU−F−FTL_RSTR, Fatal error for RESTORE operation at 13−JUL−2013 11:43:58.38
```

7.1.16 Table Names Can Now Be Specified For Index Verification

A new feature has been added to the Oracle Rdb RMU/VERIFY command which allows table names to be specified for database index verification. Currently, only a list of one or more index names can be specified with either the /INDEXES or /INDICES qualifier. Now, if one or more database table names is specified with the new /FOR_TABLE qualifier, all the indexes defined for the named table(s) will be selected for verification. If the /FOR_TABLE qualifier is used, then either the /INDEXES or /INDICES qualifier must also be specified in the same RMU/VERIFY command.

The /INDEXES or /INDICES qualifiers can continue to specify a list of one or more index names to be verified in addition to the /FOR_TABLE list of one or more table names for which all the indexes defined for each table are to be verified. Index names cannot be specified with the new /FOR_TABLE qualifier and table names cannot be specified with the existing /INDEXES or /INDICES qualifiers. Either the syntax RMU/VERIFY/INDEXES/FOR_TABLE=* or RMU/VERIFY/INDEXES/FOR_TABLE can be used to specify that all indexes defined for all database tables should be verified. If lower case charcters are not to be converted to upper case in table names, table names must be delimited with double quotes.

The following new syntax can be specified for the /FOR_TABLE qualifier.

```
/FOR_TABLE=(table_name,...)
/FOR_TABLE=table_name
/FOR_TABLE=*
/FOR_TABLE
```

The following examples show that both the /FOR_TABLE and /INDEXES or /INDICES qualifiers can specify either no values or lists of one or more table or index names in the same RMU/VERIFY command line. Only table name values can be specified with the /FOR_TABLE qualifier, and only index name values can be specified with the /INDEXES or /INDICES qualifiers. If table name values are specified, all the indexes defined for each named table will be verified.

```
$RMU/VERIFY/INDEXES/FOR_TABLE=EMPLOYEES/NOLOG MF_PERSONNEL
$RMU/VERIFY/INDEXES=(EMPLOYEES_HASH,EMP_EMPLOYEE_ID)/FOR_TABLE=COLLEGES/NOLOG
MF_PERSONNEL
$RMU/VERIFY/INDICES=EMP_EMPLOYEE_ID/FOR_TABLE=(COLLEGES,JOB_HISTORY)/NOLOG
MF_PERSONNEL
```

The following examples show that both of the following commands are equivalent and will verify all indexes defined for all tables in the database.

```
$RMU/VERIFY/INDEXES/FOR_TABLE=*/NOLOG MF_PERSONNEL
$RMU/VERIFY/INDICES/FOR_TABLE/NOLOG MF_PERSONNEL
```

The following examples shows that if the /FOR_TABLE qualifier is specified, either the /INDEXES or /INDICES qualifiers must be specified in the same RMU/VERIFY command or a fatal error will occur.

```
$RMU/VERIFY/FOR_TABLE=EMPLOYEES MF_PERSONNEL
%RMU-F-CONFLSWIT, conflicting qualifiers /FOR_TABLE and /INDEXES or /INDICES
not specified
%RMU-F-FTL_VER, Fatal error for VERIFY operation at 14-FEB-2013 14:58:47.09
```

7.1.17 New RMU/VERIFY Feature to Detect Orphan Hash Index Buckets

To ensure Oracle Rdb database integrity, a new feature has been added to the RMU/VERIFY command to detect "orphan" hash index buckets on database pages in mixed storage areas which do not belong to any existing hash index defined for the database. Orphan hash buckets are either not referenced in a SYSTEM RECORD on a mixed area database page or are not refernced by another hash bucket.

This is not a default feature but must be activated by specifying the new "/ORPHAN_INDEXES" qualifier on the command line. Orphan hash index buckets will only be reported if RMU/VERIFY is verifying one or more hash indexes as part of the database verification. For each orphan hash bucket detected, an error message will be output specifying the storage area name and physical "dbkey" address of the orphan hash bucket. The physical dbkey address specifies the storage area number, the storage area page number, and the storage area line number of the orphan hash bucket. This information can be used with the RMU/DUMP command to dump the area page where the orphan hash bucket is located.

In the short term, these orphaned hash index elements are harmless but consume space which would otherwise be used by new inserts. Eventually, as objects get dropped and created, these elements may be confused with current structures. Therefore, Oracle recommends cleaning them up as soon as practical.

However, these orphaned hash index elements can no longer be removed using the standard DROP commands in SQL. To reclaim the space used by these elements will require a DROP STORAGE AREA for the affected area. The database administrator should create a replacement storage area and use ALTER or DROP commands to move other tables and indices out of the affected storage area. Then use DROP STORAGE AREA to remove the unused area. Alternatively, you can use the SQL EXPORT DATABASE and IMPORT DATABASE commands to rebuild the whole database.

If RMU/VERIFY detects and reports any structural problems with the database pages or hash index stuctures for the area being verified, or any problems with VMS sort which is used in the process of detecting orphan hash buckets, detection of orphan hash buckets will be aborted for the area and the following error message will be output.

```
$ RMU/VERIFY/ALL/ORPHAN_INDEXES MF_PERSONNEL
%RMU−E−ORPHANERR, Error searching for orphan index nodes in area EMPIDS_LOW
```

The user should correct the reported problems and repeat the verification.

The new syntax for this feature which can be specified with the RMU/VERIFY command is the following.

```
/[NO]ORPHAN_INDEXES
```

The default is "/NOORPHAN_INDEXES" − orphan hash buckets will not be detected or reported. To activate this feature "/ORPHAN_INDEXES" must be specified.

The following example shows the diagnostic error message that will be put out by RMU/VERIFY for each orphan hash bucket found on a database page in the current storage area. Both the storage area name and the physical dbkey address of the orphan hash bucket are reported. The dbkey address in the message following the word "at" specifies the storage area number followed by the page number followed by the line number where the orphan hash bucket is located.

```
$ RMU/VERIFY/ALL/ORPHAN_INDEXES/NOERROR_LIMIT DATABASE.RDB
%RMU−E−ORPHANIDX, Orphan hashed index bucket found in area
 DATABASE_AREA at 21:2:5
%RMU−E−ORPHANIDX, Orphan hashed index bucket found in area
 DATABASE_AREA at 21:2:9
%RMU−E−ORPHANIDX, Orphan hashed index bucket found in area
DATABASE_AREA at 21:2510:32
%RMU−E−ORPHANIDX, Orphan hashed index bucket found in area
```

```
DATABASE_AREA at 21:2510:43
```

7.1.18 New COMPILE Clause for ALTER TRIGGER Statement

This release of Oracle Rdb supports a new COMPILE clause for the ALTER TRIGGER statement. This clause directs Rdb to re−compile the trigger to ensure that it is valid. If COMPILE is successful and the trigger was marked invalid, but "Can be revalidated" then the invalid flag will be cleared.

Triggers can be marked "invalid" when a procedure, function or sequence is dropped using the CASCADE clause.

The following example shows how this new clause could be used.

```
SQL> set flags 'warn_invalid';
SQL>
SQL> alter module M
cont>     drop procedure PP cascade;
~Xw: Trigger "C_INSERT" marked invalid
SQL>
SQL> show trigger C_INSERT
     C_INSERT
 Current state is INVALID
    Can be revalidated
 Source:
 c_insert
    after insert on C
    when (C.b is NULL)
        (call PP())
    for each row
SQL>
SQL> alter trigger C_INSERT
cont>     compile;
%RDB−E−NO_META_UPDATE, metadata update failed
−RDB−E−OBSOLETE_METADA, request references metadata objects that no longer exist
−RDMS−F−BAD_SYM, unknown routine symbol − PP
SQL>
SQL> alter module M
cont>     add procedure PP;
cont>     trace 'in PP';
cont> end module;
SQL>
SQL> alter trigger C_INSERT
cont>     compile;
SQL>
SQL> show trigger C_INSERT
     C_INSERT
 Source:
 c_insert
    after insert on C
    when (C.b is NULL)
        (call PP())
    for each row
SQL>
```

Note

Any trigger marked with the invalid flag will still be used by Oracle Rdb if at runtime it can be compiled successfully. However, only the COMPILE clause of the ALTER TRIGGER statement, or the COMPILE ALL TRIGGERS clause of the ALTER TABLE statement will clear the "invalid" flag.

7.1.19 New COMPILE ALL TRIGGERS Clause for ALTER TABLE Statement

This release of Oracle Rdb supports a new COMPILE ALL TRIGGERS clause for the ALTER TABLE statement. This clause directs Rdb to re−compile all the triggers defined for the table to ensure that they are valid. If COMPILE ALL TRIGGERS is successful and any trigger was marked invalid and "Can be revalidated" then the invalid flag will be cleared.

Triggers can be marked "invalid" when a procedure, function or sequence is dropped using the CASCADE clause.

The following example shows how this new clause could be used.

```
SQL> set flags 'warn_invalid';
SQL>
SQL> alter module M
cont>     drop procedure PP cascade;
~Xw: Trigger "C_INSERT" marked invalid
SQL>
SQL> show trigger C_INSERT
     C_INSERT
 Current state is INVALID
    Can be revalidated
 Source:
 c_insert
    after insert on C
    when (C.b is NULL)
       (call PP())
    for each row
SQL>
SQL> alter table C
cont>     compile all triggers;
%RDB−E−NO_META_UPDATE, metadata update failed
−RDB−E−OBSOLETE_METADA, request references metadata objects that no longer exist
−RDMS−F−BAD_SYM, unknown routine symbol − PP
SQL>
SQL> ! Replace missing procedure PP
SQL> alter module M
cont>     add procedure PP;
cont>     trace 'in PP';
cont> end module;
SQL>
SQL> alter table C
cont>     compile all triggers;
SQL>
SQL> ! Show that the INVALID flag is now cleared
SQL> show trigger C_INSERT
     C_INSERT
 Source:
 c_insert
    after insert on C
    when (C.b is NULL)
       (call PP())
    for each row
SQL>
```

Note

Any trigger marked with the invalid flag will still be used by Oracle Rdb if at runtime it can be compiled successfully. However, only the COMPILE clause of the ALTER TRIGGER statement or the COMPILE ALL TRIGGERS clause of the ALTER TABLE statement will clear the "invalid" flag.

7.1.20 New RETRY Clause for ACCEPT Statement

This release of Oracle Rdb adds a new RETRY clause to the ACCEPT Statement. The RETRY clause specifies the number of times that SQL will reprompt the user when an error occurs after processing the user's input.

The following example shows that after an erroneous input, the user is prompted again for the data.

```
SQL> declare :v integer = 0;
SQL> accept :v retry 5;
Enter value for V: xxxx
%RDB−E−ARITH_EXCEPT, truncation of a numeric value at runtime
−COSI−F−INPCONERR, input conversion error
Enter value for V: 42
SQL> print :v;
          V
         42
SQL>
```

7.1.21 New Character Sets ISOLATIN2 and WIN_LATIN2 Supported

This release of Oracle Rdb adds two new character sets, ISOLATIN2 and WIN_LATIN2.

Usage Notes

ISOLATIN2 is a single octet character set that has the following characteristics:

- Encodes Extended East European characters as defined by the ISO/IEC 8859−2 standard.
- Fixed single octet characters.
- May be used as an Identifier character set.
- Contains the full set of ASCII characters.
- EE8ISO8859P2 is the Oracle NLS equivalent character set.
- The translation name to translate to ISOLATIN2 characters is RDB$ISOLATIN2.
- The Wildcard Underscore character is %X5F.
- The Wildcard Percent character is %X25.

WIN_LATIN2 is a single octet character set that has the following characteristics:

- Encodes Extended East European characters as defined by the MS Windows Code Page 1250 8−Bit standard.
- Fixed single octet characters.
- May be used as an Identifier character set.
- Contains the full set of ASCII characters.
- EE8MSWIN1250 is the Oracle NLS equivalent character set.
- The translation name to translate to WIN_LATIN2 characters is RDB$WIN_LATIN2.
- The Wildcard Underscore character is %X5F.
- The Wildcard Percent character is %X25.

7.1.22 Changes and Enhancements to Trigger Support

In this release of Oracle Rdb, the handling of trigger definitions has been changed to allow future enhancements to triggers.

The following changes may be observed with this release.

- In prior versions, the entire definition was stored in a single row in the system table Rdb$TRIGGERS. With this release, all new trigger definitions are split into separate rows stored in Rdb$TRIGGER_ACTIONS with a single base row in Rdb$TRIGGERS.
- Each trigger action is given a generated action name.
- Each trigger is assigned a unique trigger identification.
- Trigger definitions that existed in prior releases of Oracle Rdb will remain unchanged in a database converted to Oracle Rdb Release 7.3 using RMU/CONVERT or RMU/RESTORE (which implicitly calls RMU/CONVERT). All new triggers created with Oracle Rdb Release 7.3 or later will use the new format.
- Oracle Rdb no longer supports the export of Triggers from remote databases older than Rdb V6.0. These would be older systems running on VAX and Alpha systems with Rdb V5.1 or earlier. The EXPORT DATABASE will need to be run under that Rdb release and not remotely from an Oracle Rdb V7.3 system.

It should be noted that the SQL (and RDO) EXPORT DATABASE statement will now save extended attributes. Therefore, interchange files created with Oracle Rdb V7.3 used with older versions will not support new trigger definitions. Oracle Rdb V7.2.5.3 or later is required to handle the new interchange format.

The following example shows the errors reported when triggers created using Rdb V7.3 and exported, are imported using an older Rdb V7.2 release.

```
SQL> IMPORT DATABASE FROM TP_EXP.RBR FILENAME TP2;
%SQL−F−NOTRGRES, Unable to IMPORT trigger TELLER_DELETE
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−E−BAD_CODE, corruption in the query string
%SQL−F−NOTRGRES, Unable to IMPORT trigger TELLER_INSERT
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−E−BAD_CODE, corruption in the query string
SQL>
```

Oracle recommends using RMU/EXTRACT/ITEM=TRIGGERS if the triggers need to be recreated in prior versions. RMU Extract is a best−effort utility and some manual editing of the generated SQL syntax may be required.

7.1.23 New RMU BACKUP RBF File BRHK_ROOT1, BRHK_ROOT2, BRH$K_ROOT3 Records /kroot_records

In this release of Oracle Rdb, three new BRH record types have been added to the RBF file used to back up Oracle Rdb databases. New BRHK_ROOT1, BRHK_ROOT2, and BRH$K_ROOT3 records have been added for backing up the enlarged Rdb root file KROOT structure in three parts. This will preserve the current minimum /BLOCK_SIZE of 2048 that can be specified for determining the buffer size used for backing up BRH records to the backup RBF file. For this release of Rdb, the KROOT has been enlarged to 5120 bytes which will not fit into the smaller block sizes that can be specified with the optional /BLOCK_SIZE qualifier for the RMU /BACKUP, DUMP/BACKUP, /BACKUP/AFTER_JOURNAL and /OPTIMIZE/AFTER_JOURNAL commands. Since the enlarged KROOT is now backed up and restored in three parts, the current range of between 2048 and 65,024 bytes that can be specified with the optional /BLOCK_SIZE qualifier has not changed for this release.

The BRH record type previously used to back up the smaller 1536 byte Rdb KROOT root structure for Rdb V7.2 and earlier releases in one record was BRH$K_ROOT. This record type will no longer be in RBF backup files produced by RMU BACKUP commands created by this version of Rdb, but it will be accepted by the RMU/RESTORE and RMU/DUMP/BACKUP commands which currently accept RBF files produced by previous V72, V71 and V70 versions of Oracle Rdb. However, V72 and previous versions of Oracle Rdb will not accept RBF records produced by this release but will return the error *%RMU−E−INVRECTYP, invalid record type in backup file* for the new BRHK_ROOT1, BRHK_ROOT2 and BRH$K_ROOT3 backup record types.

The following example shows the Oracle Rdb V7.3 database backup file MFP73.DMP created by the RMU/BACKUP command, which is then dumped with the most detailed "DEBUG" option by the RMU/DUMP/BACKUP command. The portion of the dump file shown contains the new BRH$K_ROOT1 (TYPE = 32), BRH$K_ROOT2 (TYPE = 33) and BRH$K_ROOT3 (TYPE = 34) records now used to backup the enlarged Rdb root file KROOT structure.

```
$ RMU/BACKUP MF_PERSONNEL.RDB MFP73.RBF
$ RMU/DUMP/BACKUP/OPTION=DEBUG/OUTPUT=MFP73.DMP MFP73.RBF
$ TYPE MFP73.DMP

REC_SIZE = 1708          REC_TYPE = 32            BADDATA = 00
ROOT = 01                AREA_ID = 0              LAREA_ID = 0
PNO = 0

00000000000000000000000A002006AC  0000   '. ..............'

KODA database root record
0000000000000049544F4F52534D4452  0000   'RDMSROOTI.......'
2EC700C900A66EB9EBCF255B00000000  0010   '.....[%Ïë¹n.É.Ç.'
00000021000000880000001700000000F  0020   '............!...'
0000000B000000280000001900000022  0030   '".......(.......'
0000007000000080000003A0000006C  0040   'l...:.......p...'
00000010000000040000001200000001  0050   '................'
00000014000000140000001000000032  0060   '2...............'
00000003000000050000000FA00000006  0070   '....ú...........'
000000000000000A0000000A0000000A  0080   '................'
00000000000000000000000000000000  0090   '................'
00000100000000000000000000000000  00A0   '................'
00000000000000000000000000000000  00B0   '................'
```

```
                                  ::::  (1 duplicate line)
00000003000000020000000200000000  00D0   '................'
000000080000000800AC93EB3CA7391C  00E0   '.9§<ë...........'
00000001000000040000000500000005  00F0   '................'
00000000000000040000002400000002  0100   '....$...........'
00000004000000010000008A00000000  0110   '................'
000000000000000000000FF00000004  0120   '................'
00000200000000000000000000000000  0130   '................'
20571AEA0000000000000200000008B  0140   '.... .......ê.W '
0000000000AC93EB216424E900AC93EB  0150   'ë...é$d!ë.......'
00000000000000000000000000000000  0160   '................'
0000008C000000000000000000000000  0170   '................'
00000000000000000000001400000010  0180   '................'
00000000000000000000000000000000  0190   '................'
00AC93EB205B2CA40000000000000000  01A0   '.........,[ ë...'
00000000000000000000000000000000  01B0   '................'
                                  ::::  (20 duplicate lines)
4F485B3A31524553555F424452455347  0300   'GSERDB_USER1:[HO'
37562E545245564E4F432E494C554843  0310   'CHULI.CONVERT.V7'
545345542E544944452E4B524F572E33  0320   '3.WORK.EDIT.TEST'
4E4E4F535245505F464D5D504C45482E  0330   '.HELP]MF_PERSONN'
0000000000000000313B4244522E4C45  0340   'EL.RDB;1........'
00000000000000000000000000000000  0350   '................'
                                  ::::  (52 duplicate lines)
        000000000000000000000000  06A0   '............'

REC_SIZE = 1706           REC_TYPE = 33             BADDATA = 00
ROOT = 01                 AREA_ID = 0               LAREA_ID = 0
PNO = 0

00000000000000000000000A002106AA  0000   'ª.!.............'

KODA database root record
00000000000000000000000000000000  0000   '................'
                                  ::::  (105 duplicate lines)
            00000000000000000000  06A0   '..........'

REC_SIZE = 1706           REC_TYPE = 34             BADDATA = 00
ROOT = 01                 AREA_ID = 0               LAREA_ID = 0
PNO = 0

00000000000000000000000A002206AA  0000   'ª.".............'

KODA database root record
00000000000000000000000000000000  0000   '................'
                                  ::::  (105 duplicate lines)
            00000000000000000000  06A0   '..........'
```

7.1.24 New Functions NUMTODSINTERVAL, NUMTOYMINTERVAL Supported

This release of Oracle Rdb adds two new functions for compatibility with the Oracle database.

- NUMTODSINTERVAL
  NUMTODSINTERVAL (n, 'interval_unit')
  NUMTODSINTERVAL converts n to an INTERVAL DAY TO SECOND value.
  The first argument, n, can be any numeric value. The value for interval_unit specifies the interval

qualifier and must resolve to one of the following string values: 'day', 'hour', 'minute', 'second'. Interval_unit is case insensitive. Leading and trailing values within the parentheses are ignored. The interval leading precision of the return is 9.

- NUMTOYMINTERVAL
  NUMTOYMINTERVAL (n, 'interval_unit')
  NUMTOYMINTERVAL converts n to an INTERVAL YEAR TO MONTH literal.
  The first argument, n, can be any numeric value. The value for interval_unit specifies the interval qualifier and must resolve to one of the following string values: 'year', 'month'. Interval_unit is case insensitive. Leading and trailing values within the parentheses are ignored. The interval leading precision of the return is 9.

Usage Notes

- This function is implicitly converted by SQL to the equivalent CAST function. Therefore, other facilities such as RMU Extract or SET FLAGS with the STRATEGY,DETAIL options will show CAST only.

```
SQL> select last_name
cont> from employees
cont> where birthday + NUMTOYMINTERVAL (20, 'year') > current_date;
Tables:
  0 = EMPLOYEES
Conjunct: (0.BIRTHDAY + CAST (20 AS INTERVAL YEAR(9))) > CURRENT_DATE
Get     Retrieval sequentially of relation 0:EMPLOYEES
0 rows selected
```

Examples

This example queries the PERSONNEL database and lists any employees who are more than 20 years older than their manager.

```
SQL> select e.last_name || e.first_name as employee,
cont>        e.birthday,
cont>        m.last_name || m.first_name as manager,
cont>        m.birthday
cont> from job_history jh, employees e, departments d, employees m
cont> where jh.employee_id = e.employee_id
cont>   and jh.job_end is null
cont>   and jh.department_code = d.department_code
cont>   and d.manager_id <> e.employee_id
cont>   and d.manager_id = m.employee_id
cont>   and e.birthday + NUMTOYMINTERVAL (20, 'year') < m.birthday;
EMPLOYEE                    E.BIRTHDAY    MANAGER                      M.BIRTHDAY
Iacobone       Eloi          1-May-1933   Stornelli     James       10-Jan-1960
Nash           Walter       19-Jan-1925   Keisling      Edward      21-Mar-1957
Hall           Lawrence     10-Jul-1933   Belliveau     Paul         9-May-1955
Clairmont      Rick         23-Dec-1924   Clarke        Karen       16-May-1950
Johnson        Bill         13-Apr-1927   Clarke        Karen       16-May-1950
5 rows selected
SQL>
```

7.1.25 RMU Dump Audit Command

When RMU/SET AUDIT is used to enable auditing for a database, Oracle Rdb writes records to the OpenVMS audit journal (for example SYS$MANAGER:SECURITY.AUDIT$JOURNAL). This command can be used to dump selected records from an OpenVMS AUDIT journal for a specific database for review.

This command is closely related to the RMU/LOAD/AUDIT command in that it reads and processes the rows from the audit journal.

Format

```
RMU/DUMP/AUDIT root-file-spec input-file-name
```

Command Qualifiers	Defaults
/BEFORE=timestamp	none
/FORMAT=formatting−options	/FORMAT=LIST
/LOG	/NOLOG
/OUTPUT=outputfile	/OUTPUT=SYS$OUTPUT
/SINCE=timestamp	none
/TYPE=(type−list)	/TYPE=ALL

Command Parameters

- root−file−spec
  The file specification for the database root file into which the table will be loaded. The default file extension is .rdb.
- input−file−name
  The input−file−name parameter is the name of the journal containing the audit record data to be dumped. The default file extension is .AUDIT$JOURNAL. You can determine the name of the security audit journal by using the DCL SHOW AUDIT/JOURNAL command.

Command Qualifiers

- Before=date−time
  Specifies the ending date and time for records extracted from the audit journal. The value is a standard OpenVMS date and time. Enclose the date in quotes if it also includes a space between the date and time fields. If omitted, then all records to the end of the journal will be dumped.
- Format=formatting−options
  This qualifier allows the database administrator to change the default format (LIST) to XML. The XML output is more useful for archiving and historical analysis.
  The following formatting options are accepted:
    - ♦ LIST − the default output displays the attribute and value on a single line. Long values will be split across multiple lines if necessary.
    - ♦ XML − formats the audit details as an XML record that can be archived.
    - ♦ CHARACTER_ENCODING_XML − adjusts the character encoding to that appropriate to the data being dumped in XML format. CHARACTER_ENCODING_XML is not compatible with the LIST keyword.
- Log
  If specified, RMU will display a summary line reporting the number of records read from the audit journal and the count of those displayed by RMU Dump.
- Since=date−time
  Specifies the starting date and time for records extracted from the audit journal. The value is a

standard OpenVMS date and time. Enclose the date in quotes if it also includes a space between the date and time fields. If omitted, then all records from the start of the journal will be dumped.
- Type=type−list
  Select different types of audit records. Values include: ALL, NONE, AUDIT, DACCESS, PROTECTION, and RMU. The list may contain negated values, such as /TYPE=(ALL,NORMU) so that some categories are removed. The default is to display all types of audit records.
- Output[=files−spec]
  Specifies the name of the file where output is sent. The default is SYS$OUTPUT. The default output file type is .LIS, if you specify a file name.

Usage Notes

- To use the RMU Dump Audit command you must have the RMU$SECURITY privilege in the root file ACL for the database whose security audit records are being loaded. If you do not have this privilege, you must have the OpenVMS SYSPRV or BYPASS privilege.
- The OpenVMS audit journal may contain data from multiple facilities in addition to Oracle Rdb and may also contain audit records for other databases. Therefore, only a subset may be read and formatted by RMU Dump.
- Each audit record is divided into packets. Each packet contains a piece of audit information. The output from RMU Dump Audit displays the type for the packet (for instance NSA$C_PKT_FINAL_STATUS), and the formatted value. If necessary, long text packets will be wrapped across multiple lines.
- Oracle Rdb uses a combination of OpenVMS types (those starting with NSA$C) and Oracle Rdb tags (those starting with RDBNSA$K). These tags are described below. Please refer to the relevant OpenVMS documentation for descriptions of the NSA$C types).
- The width of the terminal session is used to limit the lines for the output to SYS$OUTPUT.

Table 7−1 RDBNSA$K types

Type	Description
RDBNSA$K_PKT_DBNAME	File specification for the database root file
RDBNSA$K_PKT_TSN	TSN (transaction sequence number) for the user process
RDBNSA$K_PKT_DACCESS	Discretionary access privileges for the user; based on object ACL, and OpenVMS override privileges
RDBNSA$K_PKT_NEW_ACE	Result of a GRANT or REVOKE statement
RDBNSA$K_PKT_OBJ_TYPE	Type of object being altered
RDBNSA$K_PKT_OLD_ACE	Prior value before a GRANT or REVOKE statement
RDBNSA$K_PKT_OPERATION_CODE	Description of the operation
RDBNSA$K_PKT_RDB_PRIV_USED	Oracle Rdb privilege used for the operation
RDBNSA$K_PKT_RMU_ARGS	The RMU command line for the operation
RDBNSA$K_PKT_STATUS_CODE	OpenVMS condition following the operation attempt

Examples

Example 1: Dumping output of DACCESS records

The following example extracts just the DACCESS records since a known time.

```
$ RMU/DUMP/AUDIT-
    MF_PERSONNEL-
    SYS$MANAGER:SECURITY.AUDIT$JOURNAL-
    /TYPE=(NONE,DACCESS)-
    /LOG-
    /SINCE="14-MAR-2013 14:36:16.70"
   .
   .
   .
-------------------------------:
REC_SUBTYPE                    : DACCESS
RDBNSA$K_PKT_DBNAME            : _$1$DGA174:[SMITH.WORK.DB]MF_PE
                               : RSONNEL.RDB;1
NSA$C_PKT_AUDIT_NAME           : SECURITY
NSA$C_PKT_SYSTEM_ID            : 44262
NSA$C_PKT_IMAGE_NAME           : $1$DGA2:[SYS1.SYSCOMMON.][SYSEX
                               : E]SQL$73.EXE
NSA$C_PKT_PROCESS_ID           : 2B60A272
NSA$C_PKT_PROCESS_NAME         : Ian Smith
NSA$C_PKT_SYSTEM_NAME          : MALIBU
NSA$C_PKT_TIME_STAMP           : 14-MAR-2013 14:36:16.7079869
NSA$C_PKT_TERMINAL             : TNA38:
RDBNSA$K_PKT_TSN               : 0:9985
NSA$C_PKT_SUBJECT_OWNER        : [PROD,SMITH]
NSA$C_PKT_USERNAME             : SMITH
NSA$C_PKT_MESSAGE_TYPE_STR     : Attempted table access
NSA$C_PKT_OBJECT_NAME          : JOB_HISTORY
RDBNSA$K_PKT_OBJ_TYPE          : TABLE
RDBNSA$K_PKT_OPERATION_CODE    : Protection Change
RDBNSA$K_PKT_DACCESS           : SELECT,INSERT,UPDATE,DELETE,CRE
                               : ATE,ALTER,DROP,OPERATOR,DBADM,R
                               : EFERENCES
RDBNSA$K_PKT_PRIV_DESIRED      : DBCTRL
RDBNSA$K_PKT_STATUS_CODE       : Oracle Rdb privilege override
NSA$C_PKT_FINAL_STATUS         : %SYSTEM-S-NORMAL
RDBNSA$K_PKT_RDB_PRIV_USED     : DBADM
-------------------------------:
%RMU-I-DATRECREAD,  6454 data records read from input file.
%RMU-I-DATRECUNL,   4 data records unloaded.
```

Example 2: Dumping Output in XML Format

The following example extracts details for a time range in XML format.

```
$ RMU/DUMP/AUDIT -
    EVENTS_DB -
    SYS$MANAGER:SECURITY.AUDIT$JOURNAL -
    /FORMAT=XML -
    /SINCE="12-MAR-2013 16:02:01.40" -
    /BEFORE="12-MAR-2013 16:02:01.97" -
    /LOG

<?xml version="1.0" encoding="iso-8859-1"?>
<!-- RMU Dump Audit for Oracle Rdb V7.3-10 -->
<!-- Generated: 12-MAR-2013 16:10:16.15 -->
<!-- Database: _$1$DGA174:[PRODUCTION.DATABASES]EVENTS_DB.RDB;1 -->
<!-- Since: 2013-03-12T16:02:01.4000000 -->
<!-- Before: 2013-03-12T16:02:01.9700000 -->

<audit>
<audit_record type="AUDIT">
```

```
<database_name>_$1$DGA174:[PRODUCTION.DATABASES]EVENTS_DB.RDB;1</database_name>
<audit_name>SECURITY</audit_name>
    <system_id>44390</system_id>
    <image_name>DISK$VMSSYS:<SYS0.SYSCOMMON.SYSEXE>RMU73.EXE</image_name>
    <process_id>15928887</process_id>
    <process_name>DB_Admin</process_name>
    <system_name>PROD03</system_name>
    <time_stamp>2013-03-12T16:02:01.5802476</time_stamp>
    <terminal>TNA465:</terminal>
    <tsn>0</tsn>
    <subject_owner>[DBA,SMITH]</subject_owner>
    <username>SMITH       </username>
    <message_type>Auditing change</message_type>
    <rmu_command>RMU/SET AUDIT/TYPE=ALARM/START EVENTS_DB</rmu_command>
    <privilege_desired>RMU$SECURITY</privilege_desired>
    <status_code>RMU required privilege</status_code>
    <final_status>%SYSTEM-S-NORMAL</final_status>
    <rdb_privilege_used>RMU$SECURITY</rdb_privilege_used>
  </audit_record>
</audit>
<!-- 1 row unloaded -->
%RMU-I-DATRECREAD,  207 data records read from input file.
%RMU-I-DATRECUNL,   1 data records unloaded 12-MAR-2013 16:10:16.16.
$ set noverify
```

Example 3: Dumping output in LIST format

The following example extracts details for a time range in LIST format.

```
$ RMU/DUMP/AUDIT -
    EVENTS_DB -
    SYS$MANAGER:SECURITY.AUDIT$JOURNAL -
    /TYPE=(AUDIT) -
    /SINCE="12-MAR-2013 16:02:01.40" -
    /BEFORE="12-MAR-2013 16:02:01.97" -
    /LOG
REC_SUBTYPE                     : AUDIT
RDBNSA$K_PKT_DBNAME             : _$1$DGA174:[PRODUCTION.DATABASES]EVENTS_DB.RDB;1
NSA$C_PKT_AUDIT_NAME            : SECURITY
NSA$C_PKT_SYSTEM_ID             : 44390
NSA$C_PKT_IMAGE_NAME            : DISK$VMSSYS:<SYS0.SYSCOMMON.SYSEXE>RMU73.EXE
NSA$C_PKT_PROCESS_ID            : 215ECCE5
NSA$C_PKT_PROCESS_NAME          : DB_Admin
NSA$C_PKT_SYSTEM_NAME           : PROD03
NSA$C_PKT_TIME_STAMP            : 2013-03-12 16:02:01.5802476
NSA$C_PKT_TERMINAL              : TNA465:
RDBNSA$K_PKT_TSN                : 0:0
NSA$C_PKT_SUBJECT_OWNER         : [DBA,SMITH]
NSA$C_PKT_USERNAME              : SMITH
NSA$C_PKT_MESSAGE_TYPE_STR      : Auditing change
RDBNSA$K_PKT_RMU_ARGS           : RMU/SET AUDIT/TYPE=ALARM/START EVENTS_DB
RDBNSA$K_PKT_PRIV_DESIRED       : RMU$SECURITY
RDBNSA$K_PKT_STATUS_CODE        : RMU required privilege
NSA$C_PKT_FINAL_STATUS          : %SYSTEM-S-NORMAL
RDBNSA$K_PKT_RDB_PRIV_USED      : RMU$SECURITY
--------------------------------:
%RMU-I-DATRECREAD,  207 data records read from input file.
%RMU-I-DATRECUNL,   1 data records unloaded 12-MAR-2013 16:19:22.31.
$
```

7.1.26 New BIN_TO_NUM Numeric Function

The function BIN_TO_NUM converts a bit vector to its equivalent number. Each argument to this function represents a bit in the bit vector (the last argument is the least significant bit of the number). This function takes as arguments any numeric datatype. Each expression must evaluate to 0 or 1. This function returns a BIGINT value with zero scale. If any argument evaluates to NULL, then the result of the conversion is NULL. This function accepts from 1 to 64 arguments.

Syntax

```
−+−> BIN_TO_NUM ( −+−> value_expr −+−> ) −+−−>
                   |               |
                   +−−−−− , <−−−−−−+
```

Examples

The following example shows the result from using BIN_TO_NUM.

```
SQL> select bin_to_num (x, y, z), x, y, z from bin_tab order by 1;
                                  X             Y             Z
                    0             0             0             0
                    1             0             0             1
                    2             0             1             0
                    3             0             1             1
                    4             1             0             0
                    5             1             0             1
                    6             1             1             0
                    7             1             1             1
8 rows selected
SQL>
```

7.1.27 RMU /PROGRESS_REPORT and Control−T for RMU Backup and Restore

This release of Oracle Rdb adds a new feature to display the performance and progress of backup and restore operations. This feature can be activated by typing Control−T after the RMU Backup or Restore operation has been started from the command line or by adding /PROGRESS_REPORT to the RMU command line.

The use of Control−T has to be enabled at the DCL level using:

```
$ SET CONTROL=T
```

While a Control−T just displays the information once, the /PROGRESS_REPORT qualifier can be used to periodically print the information to SYS$OUTPUT in a batch job.

The /PROGRESS_REPORT qualifier will default to 60 seconds.

Example to display backup performance every 10 seconds:

```
$ RMU/BACKUP $1$DGA10:[DB]SAMPLE $1$DGA20:[BCK]SAMPLE /DISK /PROGRESS_REPORT=10
Read    18 MB ( 0%) at  18 MB/s, estimated completion time 14:10:41.15
   .
```

```
    .
    .
Read  3934 MB (99%) at  28 MB/s, estimated completion time 14:10:39.86
```

- Read n MB = raw blocks read so far
- (n%) = percent of total blocks read
- n MB/s = transfer rate since last display
- estimated completion time = recalculated using the current transfer rate

For parallel backups with the /PROGRESS_REPORT qualifier, each worker process puts out its own progress report as in the following example:

```
WORKER_001: Read   41 MB (25%) at 41 MB/s, estimated completion time 14:55:32.96
WORKER_002: Read   37 MB (25%) at 37 MB/s, estimated completion time 14:55:32.96
WORKER_001: Read   75 MB (46%) at 34 MB/s, estimated completion time 14:55:33.62
WORKER_002: Read   69 MB (47%) at 31 MB/s, estimated completion time 14:55:33.57
WORKER_001: Read  104 MB (65%) at 29 MB/s, estimated completion time 14:55:33.96
WORKER_002: Read  100 MB (67%) at 30 MB/s, estimated completion time 14:55:33.62
WORKER_001: Read  135 MB (84%) at 30 MB/s, estimated completion time 14:55:33.92
WORKER_002: Read  130 MB (88%) at 30 MB/s, estimated completion time 14:55:33.66
```

The following restrictions currently exist:

- Restores from other than disk files do not display the percentage completed nor the estimated completion time:

  ```
  $ RMU/RESTORE/NOCDD $1$MGA500:SAMPLE,$1$MGA600: /REWIND /PROG=10
  Read    72 MB at   14 MB/s
  Read   135 MB at   15 MB/s
      .
      .
      .
  Read  1441 MB at   12 MB/s
  ```

7.1.28 /[NO]SNAPSHOTS, /[NO]DATA_FILE Added to RMU/MOVE_AREA

Currently, RMU/MOVE_AREA moves or creates a new version of BOTH the storage area data (*.RDA) and snapshot (*.SNP) files. This new syntax allows moving ONLY the data area file or ONLY the snapshot area file for all or for named storage areas. /NODATA_FILE and /NOSNAPSHOTS are positional qualifiers that can be specified globally as a default and/or for one or more named storage areas. They can be specified on the command line or in an options file using the existing RMU/MOVE_AREA /OPTION=filespec qualifier.

The syntax for these qualifiers is as follows:

```
/[NO]SNAPSHOTS[=([FILE=filespec],[ALLOCATION=n])]
```

NOSNAPSHOTS does not move the storage area snapshot file(s). It only moves the data storage area file(s). SNAPSHOTS is the default. [=([FILE=filespec],[ALLOCATION=n])] cannot be specified with NOSNAPSHOTS. SNAPSHOTS[=([FILE=filespec],[ALLOCATION=n])] is an existing qualifier but now it can be negated. SNAPSHOTS as a local qualifier can override NOSNAPSHOTS as a global qualifier. NOSNAPSHOTS as a local qualifier can override SNAPSHOTS as a global qualifier.

```
/[NO]DATA_FILE
```

NODATA_FILE does not move the storage area data file(s). It only moves the snapshot storage area file(s). DATA_FILE is the default. It does not accept any values. DATA_FILE as a local qualifier can override NODATA_FILE as a global qualifier. NODATA_FILE as a local qualifier can override DATA_FILE as a global qualifier.

If NODATA_FILE is specified, the storage area data file is not moved or modified. A new version of the file will not be created. If NOSNAPSHOT is specified, the storage area snapshot file is not moved or modified. A new version of the file will not be created. Any existing RMU/MOVE_AREA qualifiers that would require an update/change to the data area file are disallowed if /NODATA_FILE is specified and any qualifiers that would require an update/change to the snapshot area file are disallowed if /NOSNAPSHOTS is specified.

Therefore, the following existing /MOVE_AREA qualifiers cannot be specified with either /NODATA_FILE or /NOSNAPSHOTS.

- /ROOT
- /BLOCKS_PER_PAGE
- /NODES_MAX
- /USERS_MAX

The following existing /MOVE_AREA qualifiers cannot be specified with /NODATA_FILE.

- /FILE
- /SPAMS
- /THRESHOLDS
- /READ_ONLY
- /READ_WRITE
- /EXTENSION

The following existing /MOVE_AREA qualifiers cannot be specified with /NOSNAPSHOTS.

- /SNAPSHOTS=(FILE=filespec)
- /SNAPSHOTS=(ALLOCATION=n)

In the following example, only the storage area snapshot files are moved for all database storage areas.

```
$ RMU/MOVE_AREA/ALL/NODATA_FILE/NOLOG/DIR=[.MOVE] MF_PERSONNEL.RDB
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
```

In the following example, only the storage area data files are moved for all database storage areas.

```
$ RMU/MOVE_AREA/ALL/NOSNAPSHOTS/NOLOG/DIR=[.MOVE] MF_PERSONNEL.RDB
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
```

In the following example, only the snapshot storage area file is moved for the EMP_INFO storage area and only the data storage area file is moved for the JOBS storage area. Note that for the JOBS storage area, /DATA_FILE did not need to be specified since it is the default.

```
$ RMU/MOVE_AREA/NOLOG MF_PERSONNEL.RDB -
  EMP_INFO /nodata_file -
           /snapshots=(file=DISK:[DIRECTORY]test_emp_info.snp), -
  JOBS  /data_file -
        /file=DISK:[DIRECTORY]test_jobs -
        /nosnapshots
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
```

In the following example, an options file is used to specify the storage areas to be moved. Only the data storage area file is moved for EMP_INFO; only the snapshot storage area file is moved for JOBS; and both the snapshot and data storage area files are moved for DEPARTMENTS. NOTE that /DATA_FILE and /SNAPSHOT are the defaults.

```
$ RMU/MOVE_AREA/NOLOG/DIR=DISK:[DIRECTORY]/OPTION=TESTMORE.OPT -
 MF_PERSONNEL.RDB
  EMP_INFO -
           /file=DISK:[DIRECTORY]test_emp_info.rda -
           /nosnapshot -
  JOBS     /nodata_file -
           /snapshot = (file=DISK:[DIRECTORY]test_jobs.snp)
  DEPARTMENTS -
              /file=DISK:[DIRECTORY]test_departments -
              /snapshot = (file=DISK:[DIRECTORY]test_departments.snp)
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
```

In the following example, the global default qualifiers designate that only the snapshot files should be moved for all storage areas. However, an options file is used to override the default for specific storage areas. Therefore, only the data storage area file is moved for EMP_INFO, only the snapshot storage area file is moved for JOBS, and both the snapshot and data storage area files are moved for DEPARTMENTS. Note that, in this case, /DATA_FILE needed to be specified in the options file to override the global specification of /NODATA_FILE but /NODATA_FILE did not have to be specified in the options file. Also /NOSNAPSHOT had to be specified in the options file to override the assumed global default of /SNAPSHOT.

```
$ RMU/MOVE_AREA/ALL/DIR=DISK:[DIRECTORY]/NOLOG/NODATA_FILE-
/OPTION=TESTMOVE.OPT  MF_PERSONNEL.RDB
  EMP_INFO /data_file -
           /file=DISK:[DIRECTORY]test_emp_info.rda -
           /nosnapshot
  JOBS     /nodata_file -
           /snapshot = (file=DISK:[DIRECTORY]test_jobs.snp)
  DEPARTMENTS -
              /data_file -
              /file=DISK:[DIRECTORY]test_departments -
              /snapshot = (file=DISK:[DIRECTORY]test_departments.snp)
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
```

7.1.29 Enhancements for Compression Support in SQL EXPORT DATABASE Command

This release of Oracle Rdb introduces support for compression to the SQL EXPORT DATABASE statement and associated decompression to the SQL IMPORT DATABASE statement.

Data compression is applied to the user data exported to the internal (interchange) format file. Table rows, null byte vector and LIST OF BYTE VARYING data is compressed using either the LZW (Lempel−Ziv−Welch) technique or the ZLIB algorithm developed by Jean−loup Gailly and Mark Adler. Table metadata (column names and attributes) are never compressed and the resulting file remains a structured interchange file.

In past releases, it was possible that table data, stored in the database with compression enabled, would be many times smaller in the database than when exported by SQL. In the database, a simple and fast RLE (run−length encoding) algorithm is used to store rows but this data is fully expanded by the EXPORT DATABASE statement. Enabling compression allows the result data file to be more compact using less disk space and permitting faster network transmission. The tradeoff is that more CPU time will be required for the compression and decompression of the data.

Changes to the SQL EXPORT DATABASE Statement

A new COMPRESSION clause has been added to the SQL EXPORT DATABASE statement. The default remains NO COMPRESSION. This clause accepts the following optional keywords: LZW, and ZLIB. The compression algorithms used are ZLIB (the default) or LZW. ZLIB allows further tuning with the LEVEL option that accepts a numeric level between 1 and 9. The default of 6 is usually a good trade off between result file size and the CPU cost of the compression.

```
-+-> NO COMPRESSION -----------------------------------------------------+->
 |                                                                       |
 +-> COMMPRESSION -+-----------------------------------------------------+
                   |                                                     |
                   +-> USING -+-> LZW -----------------------------------+-+
                              |                                           |
                              +-> ZLIB -+-------------------------------+-+
                                        |                               |
                                        +-> ( LEVEL numeric-literal ) -+
```

The following example shows the specification of the COMPRESSION clause.

```
SQL> export database
cont>    filename COMPLETE_WORKS
cont>    into COMPLETE_WORKS.RBR
cont>    compression using ZLIB (level 9)
cont> ;
```

Changes to the IMPORT DATABASE Statement

The metadata in the interchange file defines the compression algorithm to be used by the IMPORT DATABASE statement and indicates which tables were compressed by the EXPORT DATABASE statement.

Usage Notes

- Only the user data is compressed, therefore, additional compression may be applied using various third party compression tools such as ZIP. It is not the goal of SQL to replace such tools.
- Only one of LZW or ZLIB may be specified for the COMPRESSION option. The LEVEL clause may not be used with LZW compression technique.
- The generated interchange file (.rbr) can be processed using the RMU Dump Export command.
- The EXPORT DATABASE statement uses compression in multiple streams. Each table is treated as a separate compression stream as is each table's null byte vector and LIST OF BYTE VARYING columns.
- In some cases, compression may automatically be disabled. When the null byte vector or row data is small (less than 9 octets), the compression overhead would typically grow the data.

7.1.30 /[NO]PARTITIONS Qualifier Added to RMU/ANALYZE/INDEXES

A new /PARTITIONS qualifier has been added to RMU/ANALYZE/INDEXES which allows data to be collected and output for individual index partitions for sorted, sorted ranked and hashed indexes which are partitioned across multiple Oracle Rdb database storage areas. Previously, only index wide data was displayed by RMU/ANALYZE/INDEXES whether or not an index was partitioned.

The new /PARTITIONS qualifier will work with existing RMU/ANALYZE/INDEXES qualifiers. RMU/ANALYZE/INDEXES/PARTITIONS can be used with /OPTIONS=FULL and /OPTIONS=DEBUG to include data for different numbered levels of individual index partitions. RMU/ANALYZE/INDEXES/PARTITIONS can be used with /BINARY_OUTPUT to output record definition (*.RRD) and unload files (*.UNL) that can be used to load partition data records into an Oracle Rdb database table using the RMU/LOAD command.

If /PARTITIONS is not specified or if /PARTITIONS is specified and an index is not partitioned, only index wide data will be output by RMU/ANALYZE/INDEXES.

The syntax for this new qualifier is as follows:

```
/[NO]PARTITIONS
```

/NOPARTITIONS is the default.

The following example shows that only index wide data is output for index I1 in database PART_IND_DB.RDB if /PARTITIONS is not specified.

```
$ RMU/ANALYZE/INDEX PART_IND_DB I1
------------------------------------------------------------------------------

 Indices for database  - DISK:[DIRECTORY]PART_IND_DB.RDB;
 Created dd-mmm-yyyy hh:mm:ss.xxxx

------------------------------------------------------------------------------
 Index I1 for relation T1 duplicates allowed
 Max Level: 3, Nodes: 898, Used/Avail: 194731/357404 (54%), Keys: 10892,
 Records: 10000
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0

------------------------------------------------------------------------------
```

The following example shows that data for each index partition is output after the index wide data for index I1 in database PART_IND_DB.RDB if /PARTITIONS is specified. The partition names are P1 through P6 and the storage area names for the partitions are INDEX1 through INDEX6. The partition data is sorted by partition name.

```
$ RMU/ANALYZE/INDEX/PARTITION PART_IND_DB I1
------------------------------------------------------------------------------

 Indices for database  - DISK:[DIRECTORY]PART_IND_DB.RDB;
 Created dd-mmm-yyyy hh:mm:ss.xxxx

------------------------------------------------------------------------------
 Index I1 for relation T1 duplicates allowed
 Max Level: 3, Nodes: 898, Used/Avail: 194731/357404 (54%), Keys: 10892,
 Records: 10000
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0

 Partition P1 in area INDEX1
 Max Level: 1, Nodes: 1, Used/Avail: 0/398 (0%), Keys: 0, Records: 0
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0

 Partition P2 in area INDEX2
 Max Level: 1, Nodes: 1, Used/Avail: 21/398 (5%), Keys: 1, Records: 1
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0

 Partition P3 in area INDEX3
 Max Level: 1, Nodes: 1, Used/Avail: 72/398 (18%), Keys: 4, Records: 4
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0

 Partition P4 in area INDEX4
 Max Level: 2, Nodes: 9, Used/Avail: 1789/3582 (49%), Keys: 103, Records: 95
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0

 Partition P5 in area INDEX5
 Max Level: 3, Nodes: 885, Used/Avail: 192849/352230 (54%), Keys: 10784,
 Records: 9900
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0

 Partition P6 in area INDEX6
 Max Level: 1, Nodes: 1, Used/Avail: 0/398 (0%), Keys: 0, Records: 0
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0

------------------------------------------------------------------------------
```

The following example shows that data for each level within each index partition is output after the index wide data for index I1 in database PART_IND_DB.RDB if /OPTION=FULL is specified. The partition names are P1 through P6 and the storage area names for the partitions are INDEX1 through INDEX6. The levels are numbered in descending order.

```
$ RMU/ANALYZE/INDEX/PARTITION/OPTION=FULL PART_IND_DB I1
------------------------------------------------------------------------------

 Indices for database  - DISK:[DIRECTORY]PART_IND_DB.RDB;
 Created dd-mmm-yyyy hh:mm:ss.xxxx

------------------------------------------------------------------------------
 Index I1 for relation T1 duplicates allowed
 Max Level: 3, Nodes: 898, Used/Avail: 194731/357404 (54%), Keys: 10892,
     Records: 10000
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0

     Level: 3, Nodes: 1, Used/Avail: 184/398 (46%), Keys: 23, Records: 0
     Level: 2, Nodes: 24, Used/Avail: 6197/9552 (64%), Keys: 869, Records: 0
     Level: 1, Nodes: 873, Used/Avail: 188350/347454 (54%), Keys: 10000,
     Records: 10000

 Partition P1 in area INDEX1
 Max Level: 1, Nodes: 1, Used/Avail: 0/398 (0%), Keys: 0, Records: 0
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0

     Level: 1, Nodes: 1, Used/Avail: 0/398 (0%), Keys: 0, Records: 0

 Partition P2 in area INDEX2
 Max Level: 1, Nodes: 1, Used/Avail: 21/398 (5%), Keys: 1, Records: 1
```

```
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0

     Level: 1, Nodes: 1, Used/Avail: 21/398 (5%), Keys: 1, Records: 1

 Partition P3 in area INDEX3
 Max Level: 1, Nodes: 1, Used/Avail: 72/398 (18%), Keys: 4, Records: 4
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0

     Level: 1, Nodes: 1, Used/Avail: 72/398 (18%), Keys: 4, Records: 4

 Partition P4 in area INDEX4
 Max Level: 2, Nodes: 9, Used/Avail: 1789/3582 (49%), Keys: 103, Records: 95
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0

     Level: 2, Nodes: 1, Used/Avail: 56/398 (14%), Keys: 8, Records: 0
     Level: 1, Nodes: 8, Used/Avail: 1733/3184 (54%), Keys: 95, Records: 95

 Partition P5 in area INDEX5
 Max Level: 3, Nodes: 885, Used/Avail: 192849/352230 (54%), Keys: 10784,
     Records: 9900
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0

     Level: 3, Nodes: 1, Used/Avail: 184/398 (46%), Keys: 23, Records: 0
     Level: 2, Nodes: 23, Used/Avail: 6141/9154 (67%), Keys: 861, Records: 0
     Level: 1, Nodes: 861, Used/Avail: 186524/342678 (54%), Keys: 9900,
     Records: 9900

 Partition P6 in area INDEX6
 Max Level: 1, Nodes: 1, Used/Avail: 0/398 (0%), Keys: 0, Records: 0
     Duplicate nodes: 0, Used/Avail: 0/0 (0%), Keys: 0, Records: 0

     Level: 1, Nodes: 1, Used/Avail: 0/398 (0%), Keys: 0, Records: 0

------------------------------------------------------------------------------
```

The following example shows that, if /BINARY_OUTPUT is specified, a PARTIND.UNL file and a PARTIND.RRD file are created that can be used with the RMU/LOAD command to load partition data for the index I1 into an Oracle Rdb database table. The PARTIND.RRD file contains the fields RMU$PARTITION_NAME and RMU$AREA_NAME which will contain the partition name and area name for each record in the PARTIND.UNL file which contains partition specific data.

```
$ RMU/ANALYZE/INDEX/PARTITION/BINARY=(FILE=PARTIND.UNL,RECORD=PARTIND.RRD) -
 /OUTPUT=I1.OUT PART_IND_DB I1
$ TYPE PARTIND.RRD
DEFINE FIELD RMU$DATE DATATYPE IS DATE.
DEFINE FIELD RMU$INDEX_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$RELATION_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$PARTITION_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$AREA_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$LEVEL DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$FLAGS DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$USED DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$AVAILABLE DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_MAP DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_USED DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_AVAILABLE DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$KEY_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DATA_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_KEY_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_DATA_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_COMPRESSED_IKEY_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_IKEY_COUNT DATATYPE IS F_FLOATING.
DEFINE RECORD RMU$ANALYZE_INDEX.
    RMU$DATE.
    RMU$INDEX_NAME.
    RMU$RELATION_NAME.
    RMU$PARTITION_NAME.
    RMU$AREA_NAME.
    RMU$LEVEL.
    RMU$FLAGS.
    RMU$COUNT.
    RMU$USED.
    RMU$AVAILABLE.
    RMU$DUPLICATE_COUNT.
    RMU$DUPLICATE_MAP.
    RMU$DUPLICATE_USED.
    RMU$DUPLICATE_AVAILABLE.
    RMU$KEY_COUNT.
    RMU$DATA_COUNT.
    RMU$DUPLICATE_KEY_COUNT.
    RMU$DUPLICATE_DATA_COUNT.
    RMU$TOTAL_COMPRESSED_IKEY_COUNT.
    RMU$TOTAL_IKEY_COUNT.
END RMU$ANALYZE_INDEX RECORD.
```

7.1.31 /[NO]PARTITIONS Qualifier Added to RMU/ANALYZE Storage Statistics

A new /[NO]PARTITIONS[=(TABLES,INDEXES)] qualifier has been added to RMU/ANALYZE storage statistics which allows data to be collected and output for individual table and/or index partitions across multiple Oracle Rdb database storage areas. Previously, only statistics for storage areas and the logical areas contained within each storage area could be displayed. Now, if /PARTITIONS is specified, first statistics for each Oracle Rdb database storage area containing partitions is output. Then statistics for each partition logical area defined for each partitioned table is output. Finally, statistics for each partition logical area defined for each partitioned index is output.

The syntax for this new qualifier is as follows:

```
/[NO]PARTITIONS[=(TABLES,INDEXES)]
```

- /NOPARTITIONS is the default.
- /PARTITIONS=TABLES only outputs partitioned table statistics.
- /PARTITIONS=INDEXES only outputs partitioned index statistics.
- If only /PARTITIONS is specified, both partitioned table and partitioned index statistics are output.

If /PARTITIONS=TABLES is specified, only statistics for partitioned tables are output. If /PARTITIONS=INDEXES is specified, only statistics for partitioned indexes are output. If the /LAREAS qualifier is used to specify a list of names of partitioned tables and/or partitioned indexes, only statistics for those tables and/or indexes will be output. If the /LAREAS qualifier is used to specify a list of logical area identifier numbers, only those logical area partitions for partitioned tables and/or indexes will be output. IF /BINARY_OUTPUT is specified with /PARTITIONS, record definition (*.RRD) and binary unload files (*.UNL) are created that can be used to load storage area and logical area data records for partitioned tables and/or indexes into an Oracle Rdb database table using the RMU/LOAD command.

The RMU/ANALYZE/PARTITIONS qualifier for storage statistics cannot be specified with the following RMU/ANALYZE qualifiers: /PLACEMENT, /CARDINALITY, /INDEXES, /AREAS, /START, /END and /EXCLUDE. Note that a new qualifier not discussed here has been added to RMU/ANALYZE/INDEXES with the syntax /[NO]PARTITIONS for index specific statistics (see previous topic). If /LAREAS is used, it must specify a partitioned table name or index name or a logical area identifier number for a logical area defined for a partitioned table or partitioned index.

In the following example, for the PART_DB database with one partitioned table T1 and one partitioned index I1, first statistics for the storage areas containing table and index partitions are output: DATA1.RDA, INDEX1.RDA and INDEX2.RDA. Then statistics for the partitioned table T1, defined with one partition SYS_P00059 in area DATA1, is output. Then statistics for the two partitioned index I1 partitions, P1 in the INDEX1 storage area and P2 in the INDEX2 storage area, are output. Note that if /PARTITIONS=TABLES was specified for the PART_DB database, only statistics for the partitioned table T1 would be output and if /PARTITIONS=INDEXES was specified, only statistics for the partitioned index I1 would be output.

```
$ RMU/ANALYZE/PARTITIONS PART_DB

 Areas containing partitions for database − DISK:[DIRECTORY]PART_DB.RDB;1
 Created  9−JUN−2012 14:11:38.43

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

 Storage analysis for storage area: DATA1 − file: DISK:[DIRECTORY]DATA1.RDA;1
 Area_id: 2,  Page length: 1024,  Last page: 703

 Bytes free: 366675 (51%), bytes overhead: 164457 (23%)
 Spam count: 1, AIP count: 0, ABM count: 3
 Data records: 10000, bytes used: 188740 (26%)
   average length: 19, compression ratio: 0.90
   index records: 0, bytes used: 0 (0%)


−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

 Storage analysis for storage area: INDEX1 − file: DISK:[DIRECTORY]INDEX1.RDA;1
 Area_id: 3,  Page length: 1024,  Last page: 703

 Bytes free: 685993 (95%), bytes overhead: 33451 (5%)
 Spam count: 1, AIP count: 0, ABM count: 3
 Data records: 1, bytes used: 428 (0%)
   average length: 428, compression ratio: 1.00
   index records: 1, bytes used: 428 (0%)
     B−Tree: 428, Hash: 0, Duplicate: 0, Overflow: 0

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

 Storage analysis for storage area: INDEX2 − file: DISK:[DIRECTORY]INDEX2.RDA;1
 Area_id: 4,  Page length: 1024,  Last page: 703

 Bytes free: 685993 (95%), bytes overhead: 33451 (5%)
 Spam count: 1, AIP count: 0, ABM count: 3
 Data records: 1, bytes used: 428 (0%)
   average length: 428, compression ratio: 1.00
   index records: 1, bytes used: 428 (0%)
     B−Tree: 428, Hash: 0, Duplicate: 0, Overflow: 0
```

```
------------------------------------------------------------------------------
------------------------------------------------------------------------------

 Partitioned Tables for database - DISK:[DIRECTORY]PART_DB.RDB;1
 Created  9-JUN-2012 14:11:38.43

------------------------------------------------------------------------------

 Storage analysis for Partitioned Table: T1

------------------------------------

 Partition SYS_P00059 in area DATA1
 Logical area: T1 Logical area id : 59

 Larea id: 59,  Record type: 31, Record length: 26, Compressed

 Data records: 10000, bytes used: 188740 (26%)
   average length: 19, compression ratio: 0.90

------------------------------------------------------------------------------
------------------------------------------------------------------------------

 Partitioned Indexes for database - DISK:[DIRECTORY]PART_DB.RDB;1
 Created  9-JUN-2012 14:11:38.43

------------------------------------------------------------------------------

 Storage analysis for Partitioned Index: I1

------------------------------------

 Partition P1 in area INDEX1
 Logical area: I1 Logical area id : 60

 Larea id: 60,  Record type: 0, Record length: 215, Not Compressed

 Data records: 1, bytes used: 428 (0%)
   average length: 428

------------------------------------

 Partition P2 in area INDEX2
 Logical area: I1 Logical area id : 61

 Larea id: 61,  Record type: 0, Record length: 215, Not Compressed

 Data records: 1, bytes used: 428 (0%)
   average length: 428

------------------------------------------------------------------------------
```

In the following example for the PART_DB database with one partitioned table T1 and one partitioned index I1, the /LAREA qualifier is specified to name the index I1. Therefore only data for the partitioned index I1 is output.

```
$ RMU/ANALYZE/PARTITIONS=INDEXES/LAREA=I1 PART_DB

 Areas containing partitions for database - DISK:[DIRECTORY]PART_DB.RDB;1
 Created  9-JUN-2013 14:17:47.13

------------------------------------------------------------------------------

 Storage analysis for storage area: INDEX1 - file: DISK:[DIRECTORY]INDEX1.RDA;1
 Area_id: 3,  Page length: 1024,  Last page: 703

 Bytes free: 685993 (95%), bytes overhead: 33451 (5%)
 Spam count: 1, AIP count: 0, ABM count: 3
 Data records: 1, bytes used: 428 (0%)
   average length: 428, compression ratio: 1.00
   index records: 1, bytes used: 428 (0%)
     B-Tree: 428, Hash: 0, Duplicate: 0, Overflow: 0

------------------------------------------------------------------------------

 Storage analysis for storage area: INDEX2 - file: DISK:[DIRECTORY]INDEX2.RDA;1
 Area_id: 4,  Page length: 1024,  Last page: 703

 Bytes free: 685993 (95%), bytes overhead: 33451 (5%)
 Spam count: 1, AIP count: 0, ABM count: 3
 Data records: 1, bytes used: 428 (0%)
   average length: 428, compression ratio: 1.00
   index records: 1, bytes used: 428 (0%)
     B-Tree: 428, Hash: 0, Duplicate: 0, Overflow: 0

------------------------------------------------------------------------------
------------------------------------------------------------------------------

 Partitioned Indexes for database - DISK:[DIRECTORY]PART_DB.RDB;1
 Created  9-JUN-2012 14:17:47.13

------------------------------------------------------------------------------

 Storage analysis for Partitioned Index: I1

------------------------------------

 Partition P1 in area INDEX1
 Logical area: I1 Logical area id : 60

 Larea id: 60,  Record type: 0, Record length: 215, Not Compressed

 Data records: 1, bytes used: 428 (0%)
   average length: 428

------------------------------------

 Partition P2 in area INDEX2
 Logical area: I1 Logical area id : 61

 Larea id: 61,  Record type: 0, Record length: 215, Not Compressed

 Data records: 1, bytes used: 428 (0%)
   average length: 428

------------------------------------------------------------------------------
```

In the following example for the PART_DB database with one partitioned table T1 and one partitioned index I1, the /LAREA qualifier specifies the logical area identifier number "60". Therefore only data for the single partition P1 for the index I1 is output.

```
$ RMU/ANALYZE/PARTITIONS/LAREA=60 PART_DB.RDB

 Areas containing partitions for database − DISK:[DIRECTORY]PART_DB.RDB;1
 Created  9−JUN−2012 15:25:23.68

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

 Storage analysis for storage area: INDEX1 − file: DISK:[DIRECTORY]INDEX1.RDA;1
 Area_id: 3,  Page length: 1024,  Last page: 703

 Bytes free: 685993 (95%), bytes overhead: 33451 (5%)
 Spam count: 1, AIP count: 0, ABM count: 3
 Data records: 1, bytes used: 428 (0%)
   average length: 428, compression ratio: 1.00
   index records: 1, bytes used: 428 (0%)
     B−Tree: 428, Hash: 0, Duplicate: 0, Overflow: 0

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

 Partitioned Indexes for database − DISK:[DIRECTORY]PART_DB.RDB;1
 Created  9−JUN−2012 15:25:23.68

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

 Storage analysis for Partitioned Index: I1

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

 Partition P1 in area INDEX1
 Logical area: I1 Logical area id : 60

 Larea id: 60,  Record type: 0, Record length: 215, Not Compressed

 Data records: 1, bytes used: 428 (0%)
   average length: 428

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
```

The following example shows that if /BINARY_OUTPUT is specified, a PART.UNL file and a PART.RRD file are created that can be used with the RMU/LOAD command to load storage partition data into an Oracle Rdb database table. The PART.RRD file contains the new fields RMU$TABLE_NAME, RMU$INDEX_NAME, RMU$PARTITION_NAME and RMU$ST_AREA_NAME for partition specific data. Note that RMU$AREA_NAME contains the logical area name and RMU$ST_AREA_NAME contains the storage area name that contains the partition or index.

```
$ RMU/ANALYZE/PARTITIONS −
 −$ /BINARY=(FILE=PART.UNL,RECORD=PART.RRD) −
 −$ /OUTPUT=PART.OUT PART_DB
$ TYPE PART.RRD
DEFINE FIELD RMU$DATE DATATYPE IS DATE.
DEFINE FIELD RMU$AREA_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$TABLE_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$INDEX_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$PARTITION_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$ST_AREA_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$STORAGE_AREA_ID DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$FLAGS DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$TOTAL_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$EXPANDED_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$FRAGMENTED_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$EXPANDED_FRAGMENT_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$FRAGMENTED_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$FRAGMENT_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$PAGE_LENGTH DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$MAX_PAGE_NUMBER DATATYPE IS SIGNED LONGWORD.
DEFINE FIELD RMU$FREE_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$OVERHEAD_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$AIP_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$ABM_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$SPAM_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$INDEX_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$BTREE_NODE_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$HASH_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATES_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$OVERFLOW_BYTES DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$LOGICAL_AREA_ID DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$RELATION_ID DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$RECORD_ALLOCATION_SIZE DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$TOTAL_SPACE DATATYPE IS F_FLOATING.
DEFINE RECORD RMU$ANALYZE_AREA.
    RMU$DATE.
    RMU$AREA_NAME.
    RMU$TABLE_NAME.
    RMU$INDEX_NAME.
    RMU$PARTITION_NAME.
    RMU$ST_AREA_NAME.
    RMU$STORAGE_AREA_ID.
    RMU$FLAGS.
    RMU$TOTAL_BYTES.
    RMU$EXPANDED_BYTES.
    RMU$FRAGMENTED_BYTES.
    RMU$EXPANDED_FRAGMENT_BYTES.
    RMU$TOTAL_COUNT.
    RMU$FRAGMENTED_COUNT.
    RMU$FRAGMENT_COUNT.
    RMU$PAGE_LENGTH.
    RMU$MAX_PAGE_NUMBER.
    RMU$FREE_BYTES.
    RMU$OVERHEAD_BYTES.
    RMU$AIP_COUNT.
    RMU$ABM_COUNT.
    RMU$SPAM_COUNT.
    RMU$INDEX_COUNT.
    RMU$BTREE_NODE_BYTES.
    RMU$HASH_BYTES.
    RMU$DUPLICATES_BYTES.
    RMU$OVERFLOW_BYTES.
    RMU$LOGICAL_AREA_ID.
    RMU$RELATION_ID.
    RMU$RECORD_ALLOCATION_SIZE.
    RMU$TOTAL_SPACE.
END RMU$ANALYZE_AREA RECORD.
```

7.1.32 /[NO]PARTITIONS Qualifier Added to RMU/ANALYZE/PLACEMENT

The RMU/ANALYZE/PLACEMENT command collects and displays statistical information describing table row placement relative to the index structure for an Oracle Rdb database. A new /PARTITIONS qualifier has been added to RMU/ANALYZE/PLACEMENT which allows placement data to be collected and output for individual index partitions for sorted, sorted ranked and hashed indexes which are partitoned across multiple Oracle Rdb database storage areas. Previously, only index wide data was displayed by RMU/ANALYZE/PLACEMENT whether or not an index was partitioned. The new /PARTITIONS qualifier will work with existing RMU/ANALYZE/PLACEMENT qualifiers.

RMU/ANALYZE/PLACEMENT/PARTITIONS can be used with /OPTIONS=FULL and /OPTIONS=DEBUG to include histogram displays for individual index partitions. RMU/ANALYZE/PLACEMENT/PARTITIONS can be used with /BINARY_OUTPUT to output record definition (*.RRD) and unload files (*.UNL) that can be used to load partition placement data records into an Oracle Rdb database table using the RMU/LOAD command.

The syntax for this new qualifier is as follows:

```
/[NO]PARTITIONS
```

/NOPARTITIONS is the default.

If /PARTITIONS is not specified or if /PARTITIONS is specified and an index is not partitioned, only index wide placement data will be output by RMU/ANALYZE/PLACEMENT.

The following example shows that only index wide placement data is output for index I1 in database PART_IND_DB.RDB, if /PARTITIONS is not specified.

```
$ RMU/ANALYZE/PLACEMENT PART_IND_DB I1
----------------------------------------------------------------------------

 Indices for database  - DISK:[DIRECTORY]PART_IND_DB.RDB;
 Created dd-mmm-yyyy hh:mm:ss.xxxx

----------------------------------------------------------------------------
 Index I1 for relation T1 duplicates allowed
 Levels: 3, Nodes: 898, Keys: 10892, Records: 10000
   Dup nodes: 0, Dup keys: 0, Dup records: 0
 Maximum path length --  dbkeys: 4, IO range: 2 to 4
 Average path length --  dbkeys: 3.99, IO range: 3.96 to 3.96

----------------------------------------------------------------------------
```

The following example shows that placement data for each index partition is output after the index wide placement data for index I1 in database PART_IND_DB.RDB, if /PARTITIONS is specified. The partition names are P1 through P6 and the storage area names for the partitions are INDEX1 through INDEX6. The partition data is sorted by partition name.

```
$ RMU/ANALYZE/PLACEMENT/PARTITION PART_IND_DB I1
----------------------------------------------------------------------------

 Indices for database  - DISK:[DIRECTORY]PART_IND_DB.RDB;
 Created dd-mmm-yyyy hh:mm:ss.xxxx
----------------------------------------------------------------------------
 Index I1 for relation T1 duplicates allowed
 Levels: 3, Nodes: 898, Keys: 10892, Records: 10000
   Dup nodes: 0, Dup keys: 0, Dup records: 0
 Maximum path length --  dbkeys: 4, IO range: 2 to 4
 Average path length --  dbkeys: 3.99, IO range: 3.96 to 3.96


 Partition P1 in area INDEX1
 Levels: 1, Nodes: 1, Keys: 0, Records: 0
   Dup nodes: 0, Dup keys: 0, Dup records: 0
 Maximum path length --  dbkeys: 0, IO range: 0 to 0
 Average path length --  dbkeys: 0.00, IO range: 0.00 to 0.00


 Partition P2 in area INDEX2
 Levels: 1, Nodes: 1, Keys: 1, Records: 1
   Dup nodes: 0, Dup keys: 0, Dup records: 0
 Maximum path length --  dbkeys: 2, IO range: 2 to 2
 Average path length --  dbkeys: 2.00, IO range: 2.00 to 2.00


 Partition P3 in area INDEX3
 Levels: 1, Nodes: 1, Keys: 4, Records: 4
   Dup nodes: 0, Dup keys: 0, Dup records: 0
 Maximum path length --  dbkeys: 2, IO range: 2 to 2
 Average path length --  dbkeys: 2.00, IO range: 2.00 to 2.00


 Partition P4 in area INDEX4
 Levels: 2, Nodes: 9, Keys: 103, Records: 95
   Dup nodes: 0, Dup keys: 0, Dup records: 0
 Maximum path length --  dbkeys: 3, IO range: 2 to 3
 Average path length --  dbkeys: 3.00, IO range: 2.87 to 2.87


 Partition P5 in area INDEX5
 Levels: 3, Nodes: 885, Keys: 10784, Records: 9900
   Dup nodes: 0, Dup keys: 0, Dup records: 0
 Maximum path length --  dbkeys: 4, IO range: 3 to 4
 Average path length --  dbkeys: 4.00, IO range: 3.97 to 3.97


 Partition P6 in area INDEX6
 Levels: 1, Nodes: 1, Keys: 0, Records: 0
   Dup nodes: 0, Dup keys: 0, Dup records: 0
 Maximum path length --  dbkeys: 0, IO range: 0 to 0
 Average path length --  dbkeys: 0.00, IO range: 0.00 to 0.00

-----------------------------------------------------------------------------
```

The following example shows that placement data and histograms for each index partition are output after the index wide placement data and histograms for index I1 in database PART_IND_DB.RDB, if /OPTION=FULL is specified. The partition names are P1 through P6 and the storage area names for the partitions are INDEX1 through INDEX6.

```
$ RMU/ANALYZE/PLACEMENT/PARTITION/OPTION=FULL PART_IND_DB I1
-----------------------------------------------------------------------------

 Indices for database  - DISK:[DIRECTORY]PART_IND_DB.RDB;
 Created dd-mmm-yyyy hh:mm:ss.xxxx

-----------------------------------------------------------------------------
 Index I1 for relation T1 duplicates allowed
 Levels: 3, Nodes: 898, Keys: 10892, Records: 10000
```

```
  Dup nodes: 0, Dup keys: 0, Dup records: 0
Maximum path length --  dbkeys: 4, IO range: 2 to 4
Average path length --  dbkeys: 3.99, IO range: 3.96 to 3.96


                       dbkey path length vs. frequency

    6 | (0)
    5 | (0)
    4 |====================================================== (9900)
    3 | (95)
    2 | (5)
    1 | (0)


                      MAX IO path length vs. frequency

    6 | (0)
    5 | (0)
    4 |====================================================== (9624)
    3 |== (359)
    2 | (17)
    1 | (0)


                      MIN IO path length vs. frequency

    6 | (0)
    5 | (0)
    4 |====================================================== (9624)
    3 |== (359)
    2 | (17)
    1 | (0)


Partition P1 in area INDEX1
Levels: 1, Nodes: 1, Keys: 0, Records: 0
  Dup nodes: 0, Dup keys: 0, Dup records: 0
Maximum path length --  dbkeys: 0, IO range: 0 to 0
Average path length --  dbkeys: 0.00, IO range: 0.00 to 0.00



Partition P2 in area INDEX2
Levels: 1, Nodes: 1, Keys: 1, Records: 1
  Dup nodes: 0, Dup keys: 0, Dup records: 0
Maximum path length --  dbkeys: 2, IO range: 2 to 2
Average path length --  dbkeys: 2.00, IO range: 2.00 to 2.00



                       dbkey path length vs. frequency

    6 | (0)
    5 | (0)
    4 | (0)
    3 | (0)
    2 |====================================================== (1)
    1 | (0)

                      MAX IO path length vs. frequency
    6 | (0)
    5 | (0)
    4 | (0)
    3 | (0)
    2 |====================================================== (1)
    1 | (0)


                      MIN IO path length vs. frequency

    6 | (0)
    5 | (0)
    4 | (0)
    3 | (0)
    2 |====================================================== (1)
    1 | (0)



Partition P3 in area INDEX3
Levels: 1, Nodes: 1, Keys: 4, Records: 4
  Dup nodes: 0, Dup keys: 0, Dup records: 0
Maximum path length --  dbkeys: 2, IO range: 2 to 2
Average path length --  dbkeys: 2.00, IO range: 2.00 to 2.00



                       dbkey path length vs. frequency

    6 | (0)
    5 | (0)
    4 | (0)
    3 | (0)
    2 |====================================================== (4)
    1 | (0)


                      MAX IO path length vs. frequency

    6 | (0)
    5 | (0)
    4 | (0)
    3 | (0)
    2 |====================================================== (4)
    1 | (0)


                      MIN IO path length vs. frequency

    6 | (0)
    5 | (0)
    4 | (0)
    3 | (0)
    2 |====================================================== (4)
    1 | (0)



Partition P4 in area INDEX4
Levels: 2, Nodes: 9, Keys: 103, Records: 95
  Dup nodes: 0, Dup keys: 0, Dup records: 0
```

```
Maximum path length --  dbkeys: 3, IO range: 2 to 3
Average path length --  dbkeys: 3.00, IO range: 2.87 to 2.87



                       dbkey path length vs. frequency

    6 | (0)
    5 | (0)
    4 | (0)
    3 |====================================================== (95)
    2 | (0)
    1 | (0)


                      MAX IO path length vs. frequency

    6 | (0)
    5 | (0)
    4 | (0)
    3 |====================================================== (83)
    2 |======= (12)
    1 | (0)


                      MIN IO path length vs. frequency

    6 | (0)
    5 | (0)
    4 | (0)
    3 |====================================================== (83)
    2 |======= (12)
    1 | (0)



Partition P5 in area INDEX5
Levels: 3, Nodes: 885, Keys: 10784, Records: 9900
  Dup nodes: 0, Dup keys: 0, Dup records: 0
Maximum path length --  dbkeys: 4, IO range: 3 to 4
Average path length --  dbkeys: 4.00, IO range: 3.97 to 3.97



                       dbkey path length vs. frequency

    6 | (0)
    5 | (0)
    4 |====================================================== (9900)
    3 | (0)
    2 | (0)
    1 | (0)


                      MAX IO path length vs. frequency

    6 | (0)
    5 | (0)
    4 |====================================================== (9624)
    3 |= (276)
    2 | (0)
    1 | (0)
```

```
                    MIN IO path length vs. frequency

      6 | (0)
      5 | (0)
      4 |========================================================= (9624)
      3 |= (276)
      2 | (0)
      1 | (0)



 Partition P6 in area INDEX6
 Levels: 1, Nodes: 1, Keys: 0, Records: 0
   Dup nodes: 0, Dup keys: 0, Dup records: 0
 Maximum path length --  dbkeys: 0, IO range: 0 to 0
 Average path length --  dbkeys: 0.00, IO range: 0.00 to 0.00
------------------------------------------------------------------------------
```

The following example shows that, if /BINARY_OUTPUT is specified, a PARTIND.UNL file and a PARTIND.RRD file are created that can be used with the RMU/LOAD command to load partition placement data for the index I1 into an Oracle Rdb database table. The PARTIND.RRD file contains the fields RMU$PARTITION_NAME and RMU$AREA_NAME, which will contain the partition name and area name for each record in the PARTIND.UNL file. PARTIND.UNL contains partition specific data.

```
$ RMU/ANALYZE/PARTITION/PARTITION/BINARY=(FILE=PARTIND.UNL,RECORD=PARTIND.RRD) -
 /OUTPUT=I1.OUT PART_IND_DB I1
$ TYPE PARTIND.RRD
DEFINE FIELD RMU$DATE DATATYPE IS DATE.
DEFINE FIELD RMU$INDEX_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$RELATION_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$PARTITION_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$AREA_NAME DATATYPE IS TEXT SIZE IS 32.
DEFINE FIELD RMU$LEVEL DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$FLAGS DATATYPE IS SIGNED WORD.
DEFINE FIELD RMU$COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_MAP_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$KEY_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_KEY_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DATA_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$DUPLICATE_DATA_COUNT DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_KEY_PATH DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_PAGE_PATH DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$TOTAL_BUFFER_PATH DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$MAX_KEY_PATH DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$MAX_PAGE_PATH DATATYPE IS F_FLOATING.
DEFINE FIELD RMU$MIN_BUF_PATH DATATYPE IS F_FLOATING.
DEFINE RECORD RMU$ANALYZE_PLACEMENT.
    RMU$DATE.
    RMU$INDEX_NAME.
    RMU$RELATION_NAME.
    RMU$PARTITION_NAME.
    RMU$AREA_NAME.
    RMU$LEVEL.
    RMU$FLAGS.
    RMU$COUNT.
    RMU$DUPLICATE_COUNT.
    RMU$DUPLICATE_MAP_COUNT.
    RMU$KEY_COUNT.
    RMU$DUPLICATE_KEY_COUNT.
    RMU$DATA_COUNT.
    RMU$DUPLICATE_DATA_COUNT.
    RMU$TOTAL_KEY_PATH.
    RMU$TOTAL_PAGE_PATH.
    RMU$TOTAL_BUFFER_PATH.
    RMU$MAX_KEY_PATH.
    RMU$MAX_PAGE_PATH.
    RMU$MIN_BUF_PATH.
END RMU$ANALYZE_PLACEMENT RECORD.
```

7.1.33 New RMU/[NO]ASSIST Qualifier for Commands Using Tape Drives

A new qualifier has been added to improve tape handling when using RMU commands which use tape volumes. The following commands have been enhanced with the new "/[NO]ASSIST" qualifier:

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

This qualifier specifies where tape handling requests are to be sent. With 'NoAssist' these requests are sent to the current process's SYS$OUTPUT device and allows a command line user to respond to these requests interactively.

With 'Assist', the requests are sent to an operator terminal and mount commands are issued with assistance enabled (see MOUNT/ASSIST).

The default for an interactive process (which can be deterined by using F$MODE()) is 'NoAssist' and for any other process is 'Assist' (for example: a batch job).

7.1.34 New RMU/ALTER Feature to Modify the Area Header Root File Specification

The Oracle Rdb RMU/ALTER user could change the file specification of the storage area and snapshot files in the Rdb database root file using the DISPLAY FILE and DEPOSIT FILE commands. He could also change the root file specification in the database root file using the DEPOSIT ROOT SPECIFICATION and DISPLAY ROOT SPECIFICATION commands. However, the RMU/ALTER user previously could not change the database root file specification contained in the storage area file and snapshot file header block which identifies the database that the storage area or snapshot file belongs to. This enhancement adds this functionality.

The new syntax, which can only be used at the RMU/ALTER commands's "RdbALTER>" prompt, is the following, where "name" is the storage area name, and "id" is the storage area identification number in the database root file.

```
DISPLAY AREA_HEADER {name|id} [SNAPSHOT] SPECIFICATION
```

This command displays the current full root file specification in the storage area file or snapshot file header block for the storage area with the specified name or number.

```
DEPOSIT AREA_HEADER {name|id} [SNAPSHOT] SPECIFICATION
[=DEV:[DIR]root_file_spec.rdb;version]
```

If the root file is not specified, this command deposits the current full root file specification in the database root in the storage area or snapshot file header block for the storage area with the specified name or number.

If the root file is specified, this command deposits the specified full database root file specification "DEV:[DIR]root_file.rdb;1" in the storage area or snapshot header block for the storage area with the specified name or number.

Any specified root file must exist or must have been changed by a previous RMU/ALTER DEPOSIT ROOT SPECIFICATION command. Only full VMS root file specifications are valid and must include a device, directory, extension and version number. Any changes to the area file headers will only be written to the actual area files when the "COMMIT" command is executed at the "RdbALTER>" prompt. Any changes to area file headers since the last "COMMIT" command was issued can be undone by executing the "ROLLBACK" command at the "RdbALTER>" prompt. "COMMIT" and "ROLLBACK" are existing RMU/ALTER commands and affect any current uncommitted changes made in RMU/ALTER, not just changes to the storage area header files.

This new feature only allows modification of the root file specification in area headers, not other area header data. The "DISPLAY AREA_HEADER" command can be used with single file databases but the root file specification will always be blank, which is standard for single file databases. The "DEPOSIT AREA_HEADER" command cannot be used with single file databases since a root file specification is not specified in area headers in single file databases. To use either the DISPLAY or DEPOSIT AREA_HEADER command, the user must be attached to the database which the areas belong to, either by specifying the database name when issuing the RMU/ALTER command or by executing the "ATTACH" command from the "RdbALTER>" prompt.

The RMU/ALTER command should be used with caution and by those familiar with the internal structure of Oracle Rdb databases. All necessary interrelated changes need to be made to the database root file, the storage area and snapshot files, and the After Image Journaling files, etc, or the database will be corrupt. A full RMU/BACKUP of the database should be done previous to invoking RMU/ALTER and a full RMU/VERIFY of the database should be done once the RMU/ALTER changes have been commited and RMU/ALTER is exited.

The following example shows that RMU/ALTER is invoked specifying the multi−file database MULTIFILE_DB.RDB which has been previously backed up using RMU/BACKUP. The DEPOSIT ROOT SPECIFICATION command is used to change the full root file specification in the root file to DEVICE:[DIRECTORY]NEW_ROOT.RDB;1. Then the DEPOSIT AREA_HEADER command is used without a root file specification to change the root file specification in the storage area header to the current root file specification set by the previous DEPOSIT ROOT SPECIFICATION command for both the ST_AREA.RDA and ST_AREA.SNP files. Then the DEPOSIT AREA_HEADER command is used with a full root file specification "DEVICE:[DIRECTORY]NEW_ROOT.RDB;1" for both the ST_AREA.RDA and ST_AREA.SNP files. This just repeats the previous change but is included to show the use of the DEPOSIT AREA_HEADER COMMAND with and without an explicit root file specification. Then a COMMIT command is used to write the changes to the database files. After exiting RMU/ALTER, the name of the old root file is changed to DEVICE:[DIRECTORY]NEW_ROOT.RDB;1 from the VMS prompt. Then the RMU/VERIFY command is used to verify the integrity of the database. The DISPLAY AREA_HEADER command is used throughout to see the current root file specification in the storage area or snapshot file headers. In the display output "(marked)" means a change has been made but has not yet been committed. Note that, although this is not shown, in this case the area headers of all storage area and snapshot files in the database need to be changed to contain the new root file specification.

```
$ rmu/alter device:[directory]multifile_db
%RMU-I-ATTACH, now altering database "DEVICE:[DIRECTORY]MULTIFILE_DB.RDB;1"

RdbALTER>  deposit root specification = DEVICE:[DIRECTORY]NEW_ROOT.RDB;1
         Root file specification is: "DEVICE:[DIRECTORY]NEW_ROOT.RDB;1"

RdbALTER> display root specification
(marked) Root file specification is: "DEVICE:[DIRECTORY]NEW_ROOT.RDB;1"

RdbALTER> display area_header st_area specification
Area ST_AREA:
         Root file specification is: "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1"

RdbALTER> display area_header st_area snapshot specification
Area ST_AREA:
         Root file specification is: "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1"

RdbALTER> deposit area_header st_area specification
Area ST_AREA:
(marked) Root file specification is: "DEVICE:[DIRECTORY]NEW_ROOT.RDB;1"

RdbALTER> deposit area_header st_area snapshot specification
Area ST_AREA:
(marked) Root file specification is: "DEVICE:[DIRECTORY]NEW_ROOT.RDB;1"

RdbALTER> deposit area_header st_area specification =
 DEVICE:[DIRECTORY]NEW_ROOT.RDB;1
Area ST_AREA:
(marked) Root file specification is: "DEVICE:[DIRECTORY]NEW_ROOT.RDB;1"

RdbALTER> deposit area_header st_area snapshot specification =
 DEVICE:[DIRECTORY]NEW_ROOT.RDB;1
Area ST_AREA:
(marked) Root file specification is: "DEVICE:[DIRECTORY]NEW_ROOT.RDB;1"

RdbALTER> commit
RdbALTER> exit

$ RENAME DEVICE:[DIRECTORY]MULTIFILE_DB.RDB;1 DEVICE:[DIRECTORY]NEW_ROOT.RDB;1
$ RMU/VERIFY/ALL DEVICE:[DIRECTORY]NEW_ROOT.RDB
```

7.1.35 REVERSE Index

Bug 5710904

This release of Oracle Rdb introduces a new sorted index attribute "reverse". A "reverse" key index reverses the bits of the key value before entering it in the index. Conceptually, a key value 24538 would become 83542 in the index (in reality, the bits of the key are reversed as opposed to the bytes). Reversing the key value can be particularly useful for indexing data such as sequence numbers, where each new key value is greater than the prior value (for example: values monotonically increase). This, in turn, can help distribute access within the index among the leaf nodes rather than concentrating the access on the "lower right corner" of the index.

Reverse key indexes may be helpful in several situations including:

- High volume transaction processing systems where they can help reduce contention for index nodes
- Applications that delete data that is older on average (with lower values of the sequence) before deleting newer data because in traditional b−trees, many index nodes may end up containing few values, with a commensurate increase in unused space

A "reverse" key index can be used for direct key lookup in a similar fashion to hash indexes. Range scans are not applicable to "reverse" key indexes.

7.1.36 Support for New Syntax for Sequence Generator Statements

This release of Oracle Rdb enhances the support for CREATE SEQUENCE, ALTER SEQUENCE and the IDENTITY clause by adding features from the ANSI and ISO SQL Language Standard.

- Alternate keywords supported in the ALTER SEQUENCE and CREATE SEQUENCE statements
  The original Rdb implementation used single keywords for negated items: NOMAXVALUE, NOMINVALUE, NOCYCLE, NOORDER, NORANDOM, and NOWAIT. However, in the SQL Standard that was published after Oracle Rdb was released, SQL uses a separate NO keyword. Rdb now supports both formats.
  The following are equivalent clauses: NOMAXVALUE and NO MAXVALUE, NOMINVALUE and NO MINVALUE, NOCYCLE and NO CYCLE, NOORDER and NO ORDER, NORANDOM and NO RANDOM, NOWAIT and NO WAIT.
- Support for IDENTITY column creation
  IDENTITY can now be followed by a list of sequence attributes: START WITH, INCREMENT BY, MAXVALUE, NO MAXVALUE, MINVALUE, NO MINVALUE, CYCLE, NO CYCLE, ORDER, and NO ORDER.
  The previous numeric list that represented a starting with and an optional increment value is supported for backward compatibility.
  Any IDENTITY created in Oracle Rdb Release 7.3 or later will default to a NO CYCLE sequence, unlike the default in prior releases. If CYCLE is desired, then include the CYCLE clause as part of the IDENTITY specification.

  ```
  SQL> create table SAMPLE1
  cont>     (a integer identity (cycle minvalue 100 no maxvalue cache 2000)
  cont>     ,b integer
  cont>     );
  SQL>
  ```
- Support for IDENTITY clause source
  SQL now captures the source for the IDENTITY clause and therefore SHOW TABLE includes more details than previous versions.

  ```
  SQL> show table (column) SAMPLE1;
  Information for table SAMPLE1

  Columns for table SAMPLE1:
  Column Name                     Data Type        Domain
  −−−−−−−−−−−                     −−−−−−−−−        −−−−−−
  ```

```
A                               INTEGER
 Computed:      Identity (cycle minvalue 100 no maxvalue cache 2000)
B                               INTEGER

SQL>
```

As in prior versions, you can use the SHOW SEQUENCE command with the name of the table to show details of the identity sequence.

```
SQL> show sequence SAMPLE1;
     SAMPLE1
 Sequence Id: 4
 An identity column sequence.
 Initial Value: 100
 Minimum Value: 100
 Maximum Value: (none)
 Next Sequence Value: 100
 Increment by: 1
 Cache Size: 2000
 No Order
 Cycle
 No Randomize
 Wait
 Comment:       column IDENTITY sequence
SQL>
```

7.1.37 RMU/SET AUDIT Supports MODULE, ROUTINE, SEQUENCE and VIEW

This release of Oracle Rdb adds several enhancements to the RMU/SET AUDIT command. The /ENABLE=DACCESS and /DISABLE=DACCESS qualifiers now accept the following new keywords for auditing.

- SEQUENCE − The SEQUENCE keyword specifies the names of sequences, either explicitly, as in this command:

  ```
  $ RMU/SET AUDIT/ENABLE=DACCESS=SEQUENCE:NEW_DEPT MF_PERSONNEL
  ```

  or via a wildcard as in this command:

  ```
  $ RMU/SET AUDIT/ENABLE=DACCESS=SEQUENCE:*ID*
  ```

  Only user defined sequences will be selected by this type of wildcard command. Those created by Oracle Rdb must be completely specified.
- ROUTINE − The ROUTINE keyword specifies the names of functions and procedures, either explicitly, as in this command:

  ```
  $ RMU/SET AUDIT/ENABLE=DACCESS=ROUTINE:CHECKSUM13 ACCOUNTING
  ```

or via a wildcard as in this command:

```
$ RMU/SET AUDIT/ENABLE=DACCESS=ROUTINE:CHECKSUM%%
```

Only user defined routines will be selected by this type of wildcard command. Those created by Oracle Rdb must be completely specified.

- MODULE − The MODULE keyword specifies the names of modules, either explicitly, as in this command:

  ```
  $ RMU/SET AUDIT/ENABLE=DACCESS=MODULE:PROD_SUPPORT MF_PERSONNEL
  ```

  or via a wildcard as in this command:

  ```
  $ RMU/SET AUDIT/ENABLE=DACCESS=MODULE:UTIL*
  ```

  Only user defined modules will be selected by this type of wildcard command. Those created by Oracle Rdb must be completely specified. Note also that routines defined with the clause USAGE IS LOCAL will not be selected by wildcards. Such routines can only be activated by routines in the same module.
  The MODULE keyword provides a time saving shortcut for auditing related routines. In a similar way that routine access control is inherited from the containing module, audit and alarm settings are inherited from the owning module when a routine is first referenced. When a subsequent ALTER MODULE ... ADD FUNCTION, or ALTER MODULE ... ADD PROCEDURE statement is used, these new routines will also inherit audit and alarm settings from the module.
- VIEW − The VIEW keyword specifies the names of views, either explicitly, as in this command:

  ```
  $ RMU/SET AUDIT/ENABLE=DACCESS=VIEW:CURRENT_JOB MF_PERSONNEL
  ```

  or via a wildcard as in this command:

  ```
  $ RMU/SET AUDIT/ENABLE=DACCESS=VIEW:CURRENT*
  ```

  Only user defined views will be selected by this type of wildcard command. Views created by Oracle Rdb must be completely specified.
  In prior releases, views could be marked for audit using the TABLE keyword. The TABLE keyword remains a superset of VIEW and selects both table and view objects. However, to perform wildcard selection of views you must use the VIEW keyword.

Note

At this time RMU/SET AUDIT accesses multischema databases using MULTISCHEMA IS OFF. Therefore, the external (possibly generated) names must be specified for the /ENABLE=DACCESS and /DISABLE=DACCESS qualifiers.

7.1.38 RMU/SHOW AUDIT Supports MODULE, ROUTINE, SEQUENCE and VIEW

This release of Oracle Rdb adds several enhancements to the RMU/SHOW AUDIT command.

The /DACCESS qualifier now accepts the following new keywords for auditing.

- SEQUENCE
  The SEQUENCE keyword reports those sequences that have audit and/or alarm settings.
- ROUTINE
  The ROUTINE keyword reports those functions and procedures that have audit and/or alarm settings. Note that routines defined within a module may inherit audit and alarm settings from the containing module and will not be reported in such cases.
- MODULE
  The MODULE keyword reports those modules that have audit and/or alarm settings.
- VIEW
  The VIEW keyword reports those views that have audit and/or alarm settings. Note that the TABLE keyword reports both tables and views that have auditing enabled.

In addition to these changes, the /DACCESS=TABLE option shows that the audited relation is a view or a table.

Note

At this time, RMU/SHOW AUDIT accesses multischema databases using MULTISCHEMA IS OFF. Therefore, the external (possibly generated) names will be displayed for all objects.

7.1.39 SQL Now Supports SQL Standard Syntax for SET CONSTRAINTS ALL Statement

This release of Oracle Rdb now supports the ANSI/ISO SQL Standard statement SET CONSTRAINTS in addition to the older Rdb syntax.

```
SET -+-> ALL CONSTRAINTS -+-----+--+-> DEFERRED ---+-->
     |                          |  |               |
     +-> CONSTRAINT --+-> ALL -+  +-> IMMEDIATE --+
     |                |           |               |
     +-> CONSTRAINTS -+           +-> DEFAULT ----+
                                  |               |
                                  +-> ON ---------+
                                  |               |
                                  +-> OFF --------+
```

The existing SET ALL CONSTRAINTS statement is retained for backward compatibility.

Please see the Oracle SQL/Services Release 7.3.1.1 Release Notes, Note 4.3.1 SET CONSTRAINTS Command Now Translated to Oracle Rdb Format, for more information.

This syntax has been implemented in Oracle Rdb Release 7.3.1.0.

7.1.40 Support ANSI and ISO SQL Standard Length Units

This release of Oracle Rdb allows the specification of the length used by data type definitions and string handling functions. In prior releases, the SET CHARACTER LENGTH statement had to precede the CREATE, DECLARE, or ALTER data definition (DDL) statements, or any usage of the SUBSTRING function in a data manipulation (DML) statement to effect the choice of character or octet units for string length and position values.

The following functions and data types are affected by this change.

- The SUBSTRING function now includes an optional USING clause to specify that either OCTETS or CHARACTERS units are used for the FROM and FOR clauses.

  ```
  SUBSTRING ( char-value-expr
              FROM start-position
              [ FOR string-length ]
              [ USING { CHARACTERS | OCTETS } ] )
  ```
- The new OVERLAY function includes an optional USING clause to specify that either OCTETS or CHARACTERS units are used for the FROM and FOR clauses.

  ```
  OVERLAY ( char-value-expr
            PLACING char-value-expr
            FROM start-position
            [ FOR string-length ]
            [ USING { CHARACTERS | OCTETS } ] )
  ```
- CHARACTER, NATIONAL CHARACTER (NCHAR), CHARACTER VARYING (VARCHAR), NATIONAL CHARACTER VARYING (NCHAR VARYING) and related types now accept an optional OCTETS or CHARACTERS option, as in the following example.

  ```
  CHAR (20 CHARACTERS)
  NATIONAL CHARACTER VARYING (300 OCTETS)
  ```
- The SIZE IS clause of the CREATE INDEX statement also accepts an optional OCTETS or CHARACTERS option.

  ```
  SIZE IS <n> [ CHARACTERS | OCTETS ]
  ```

Using these clauses will override any current setting of the SET CHARACTER LENGTH statement or the SET DIALECT statement.

The following shows examples of these changes.

```
SQL> create database
cont>     filename SAMPLE_DB
cont>     national character set DEC_KANJI
cont> ;
SQL>
SQL> set character length 'characters';
SQL>
SQL> create table EMPLOYEES
cont>     (name       nchar(10 characters)
cont>     ,department char(10 octets)
cont>     ,comments   character varying (300 characters)
cont>     );
```

```
SQL>
SQL> create index EMPLOYEES_COMMENTS
cont>     on EMPLOYEES (comments size is 30 characters)
cont> ;
SQL>
SQL> declare :fl char(20 octets);
SQL> select rdb$flags into :fl from rdb$database;
SQL> print :fl;
 FL
 131328
SQL>
SQL> select name
cont>      ,department
cont>      ,substring (comments from 1 for 30 using characters) as part_comment
cont>  from EMPLOYEES
cont>  where comments starting with 'Review:';
0 rows selected
SQL>
SQL> commit;
SQL>
```

7.1.41 New SET FLAGS Clause Supported by CREATE and ALTER PROFILE

In this release of Oracle Rdb, the CREATE and ALTER PROFILE statements have been enhanced with a SET FLAGS clause. This new clause is related to the SET FLAGS statement; refer to that documentation for the list of available keywords that can be specified.

The string associated with the SET FLAGS clause is saved with the created profile. Any user that has this assigned profile will implicitly execute SET FLAGS during session start.

Note

Please notice that some SET FLAGS keywords affect actions during database attach and so have no action when defined within a profile. For example, DATABASE_PARAMETERS, generates minimal effects in such cases.

The following example shows the creation of a profile with flags and the assigning of the profile to a user.

```
SQL> create profile SF_USER
cont> set flags
cont> 'old_cost_model,noindex_column_group,optimization_level(total_time)'
cont> ;
SQL>
SQL> alter user SMITH
cont> profile SF_USER;
SQL>
```

When this user (SMITH) attaches to a database (ATTACH, CONNECT, DECLARE ALIAS), or uses SET SESSION AUTHORIZATION, a SET FLAGS statement will implicitly be executed using this string of keywords.

Note that the CREATE PROFILE and ALTER PROFILE statements will validate the listed keywords.

```
SQL> alter profile SF_USER
cont> set flags 'THIS_OLD_HOUSE'
cont> ;
%RDB−E−NO_META_UPDATE, metadata update failed
−RDB−E−EXT_ERR, Oracle Rdb extension error
−RDMS−E−UNKAMBFLAG, 'THIS_OLD_HOUSE' is an unknown or ambiguous flag name
SQL>
```

The clause NO SET FLAGS can be used to remove any flags associated with the profile. All users assigned that profile will no longer perform a SET FLAGS action during session start.

```
SQL> show profile SF_USER
     SF_USER
     Flags: "old_cost_model,noindex_column_group,optimization_level(total_time)"
SQL> alter profile SF_USER
cont> no set flags;
SQL> show  profile SF_USER
     SF_USER
SQL>
```

7.1.42 New Support for SAVEPOINT Syntax and Semantics

This release of Oracle Rdb adds support for a SAVEPOINT. The SAVEPOINT feature allows the programmer to place a marker within a transaction that can later be used to undo part of the transaction using the ROLLBACK TO SAVEPOINT statement. Additionally, this marker can be freed using the RELEASE SAVEPOINT statement.

Note

At this time, Oracle Rdb only supports a single active SAVEPOINT per transaction.

Some SQL DDL statements currently use this feature to implement SQL semantics and therefore mixing SAVEPOINT and these statements is not supported. Some SQL statements that use SAVEPOINT include: GRANT and REVOKE using * wildcard object names, SET CONSTRAINT MODE 'ON', some forms of ALTER MODULE statement, and an INSERT ... SELECT statement that uses two database aliases.

7.1.42.1 SAVEPOINT Statement

The SAVEPOINT statement establishes a marker in the current transaction that allows the programmer to undo part of the transaction (using ROLLBACK TO SAVEPOINT) without resorting to a full transaction ROLLBACK.

Syntax

```
SAVEPOINT −+−−−−−−−−−−−−−−−−−+−> savepoint−name −+−>
           |                 |
           +−> alias−name . −+
```

Arguments

- alias−name
  This optional alias name can be used to target a specific database alias. If no alias−name is provided,

then the current default database will be used.

- savepoint−name
  Name of a unique identifier for this savepoint. This name will be used with subsequent ROLLBACK TO SAVEPOINT and RELEASE SAVEPOINT statements.

Usage Notes

- If the SAVEPOINT statement is used more than once with the same name, then the prior SAVEPOINT is destroyed and replaced with this new location.
- Any established savepoints will be discarded by a ROLLBACK statement (which does not use the TO SAVEPOINT clause), and by a COMMIT statement.
- If more savepoints are created than are supported by Rdb, then the error RDB$_EXCESS_SVPT will be raised. SQLCODE will be returned as −880 and SQLSTATE will be returned as 3B002.

  ```
  %RDB−E−EXCESS_SVPT, maximum number of savepoints are already active −
  "BOOK2" failed
  ```
- The SAVEPOINT statement may not be used in a SQL function definition nor can it be called indirectly from a function.
- The SAVEPOINT statement may not be called indirectly from a trigger action.
- A SAVEPOINT statement is only valid if a transaction is in progress. This can be either a READ WRITE or READ ONLY transaction. Note that temporary tables can be updated during a read only transaction.

  ```
  SQL> commit;
  SQL> savepoint BK;
  %RDB−E−NOTXNINPRGS, no transaction is in progress
  −RDB−E−SVPT_NOALLOWED, a savepoint may not be established in this context −
  "BK" failed
  ```

The following example shows the use of the SAVEPOINT statement. Note that reusing the savepoint name will re−establish that marker and so affect different rows in the transaction.

```
SQL> declare local temporary table module.SAMPLE
cont>     (a integer)
cont>     on commit preserve rows
cont> ;
SQL>
SQL> --
SQL>
SQL> set transaction read only;
SQL>
SQL> insert into module.SAMPLE values (1);
1 row inserted
SQL>
SQL> -- Establish the initial marker
SQL> savepoint BOOK_IT;
SQL>
SQL> insert into module.SAMPLE values (2);
1 row inserted
SQL> insert into module.SAMPLE values (3);
1 row inserted
SQL>
SQL> table module.SAMPLE;
           A
           1
           2
           3
```

```
3 rows selected
SQL>
SQL> -- Move the marker
SQL> savepoint BOOK_IT;
SQL>
SQL> insert into module.SAMPLE values (4);
1 row inserted
SQL>
SQL> rollback to savepoint BOOK_IT;
SQL>
SQL> table module.SAMPLE;
           A
           1
           2
           3
3 rows selected
SQL>
SQL> commit;
SQL>
```

7.1.42.2 RELEASE SAVEPOINT Statement

The RELEASE SAVEPOINT Statement destroys the named savepoint established by the SAVEPOINT statement. Changes made by the transaction are unaffected by this statement.

Syntax

```
RELEASE SAVEPOINT -+-----------------+-> savepoint-name -+->
                   |                 |
                   +-> alias-name . -+
```

Arguments

- alias−name
  This optional alias name can be used to target a specific database alias. If no alias−name is provided, then the current default database will be used.
- savepoint−name
  Name of a unique identifier for this savepoint. This name is declared using the SAVEPOINT statement.

Usage Notes

- If no established savepoint exists with this name, then the error RDB$_BAD_SVPT_HANDLE will be raised. SQLCODE will be returned as −882 and SQLSTATE will be returned as 3B001.

  ```
  %RDB-E-BAD_SVPT_HANDLE, invalid savepoint handle - "BOOKMARK2" is unknown
  ```
- The RELEASE SAVEPOINT statement may not be used in a SQL function definition nor can it be called indirectly from a function.
- The RELEASE SAVEPOINT statement may not be called indirectly from a trigger action.

The following example shows the use of the RELEASE SAVEPOINT statement.

```
SQL> set transaction read write;
SQL>
```

```
SQL> insert into module.SAMPLE values (1);
1 row inserted
SQL>
SQL> savepoint BOOK_IT;
SQL>
SQL> insert into module.SAMPLE values (2);
1 row inserted
SQL> insert into module.SAMPLE values (3);
1 row inserted
SQL>
SQL> table module.SAMPLE;
           A
           1
           2
           3
3 rows selected
SQL>
SQL> release savepoint BOOK_IT;
SQL>
SQL> insert into module.SAMPLE values (4);
1 row inserted
SQL>
SQL> table module.SAMPLE;
           A
           1
           2
           3
           4
4 rows selected
SQL>
SQL> commit;
SQL>
```

7.1.42.3 ROLLBACK TO SAVEPOINT Statement

The ROLLBACK TO SAVEPOINT statement destroys the named savepoint established by the SAVEPOINT statement and removes all database changes made from the time the SAVEPOINT statement established the named savepoint.

Syntax

```
ROLLBACK TO −+−−−−−−−−−−−−−+−−+−−−−−−−−−−−−−−−−−+−> savepoint−name −+−>
             |             |  |                 |
             +−> SAVEPOINT −+  +−> alias−name . −+
```

Arguments

- alias−name
  This optional alias name can be used to target a specific database alias. If no alias−name is provided then the current default database will be used.
- savepoint−name
  Name of a unique identifier for this savepoint. This name is declared using the SAVEPOINT statement.

Usage Notes

- If no established savepoint exists with this name then the error RDB$_BAD_SVPT_HANDLE will be raised. SQLCODE will be returned as −882 and SQLSTATE will be returned as 3B001.

  ```
  %RDB−E−BAD_SVPT_HANDLE, invalid savepoint handle − "BOOKMARK2" is unknown
  ```
- The ROLLBACK TO SAVEPOINT statement may not be used in a SQL function definition nor can it be called indirectly from a function.
- The ROLLBACK TO SAVEPOINT statement may not be called indirectly from a trigger action.

The following example shows the use of SAVEPOINT and ROLLBACK TO SAVEPOINT to exclude rows inserted during the transaction. In an actual application, the ROLLBACK TO SAVEPOINT statement would probably be within a conditional statement such as IF−THEN−ELSE or CASE statement.

```
SQL> declare local temporary table module.SAMPLE
cont>     (a integer)
cont>     on commit preserve rows
cont> ;
SQL>
SQL> set transaction read only;
SQL>
SQL> insert into module.SAMPLE values (1);
1 row inserted
SQL>
SQL> savepoint BOOK_IT;
SQL>
SQL> insert into module.SAMPLE values (2);
1 row inserted
SQL> insert into module.SAMPLE values (3);
1 row inserted
SQL>
SQL> table module.SAMPLE;
           A
           1
           2
           3
3 rows selected
SQL>
SQL> rollback to savepoint BOOK_IT;
SQL>
SQL> insert into module.SAMPLE values (4);
1 row inserted
SQL>
SQL> table module.SAMPLE;
           A
           1
           4
2 rows selected
SQL>
SQL> commit;
SQL>
```

7.1.43 New OPTIMIZE OUTLINE Clause Allows Outline Specification

Bug 2835544

This release of Oracle Rdb extends the use of query outlines so they can be specified in−line with SELECT, DELETE, UPDATE, INSERT and BEGIN PRAGMA statements.

In some cases, using the CREATE OUTLINE statement to define a query outline might not be possible or permitted. The OUTLINE option is part of the OPTIMIZE clause and allows the query outline to be packaged with the query.

The following example shows a query modified with an outline.

```
SQL> set flags 'strategy,detail(2),request_name';
SQL> select last_name, middle_initial, first_name
cont> from employees2
cont> where last_name = 'Toliver' and first_name = 'Alvin'
cont> optimize
cont>     as test3
cont>     outline (
cont>         mode 0
cont>         as (
cont>           query (
cont>             subquery (
cont>               EMPLOYEES2 0  access path index  E3_INDEX
cont>               )
cont>             )
cont>           )
cont>         compliance optional
cont>         execution options (total time)
cont>         );
~Query Name: "TEST3"
~S: Outline "(unnamed)" used
Tables:
  0 = EMPLOYEES2
Leaf#01 BgrOnly 0:EMPLOYEES2 Card=100
  Bool: (0.LAST_NAME = 'Toliver') AND (0.FIRST_NAME = 'Alvin')
  BgrNdx1 E3_INDEX [1:1] Fan=14
    Keys: 0.LAST_NAME = 'Toliver'
 LAST_NAME        MIDDLE_INITIAL   FIRST_NAME
 Toliver          A.               Alvin
1 row selected
SQL>
```

This example was produced by using the output from the SET FLAGS 'OUTLINE' statement and capturing the portion that defines the outline actions.

Usage Notes

- If MODE is specified with the OPTIMIZE OUTLINE clause, then the specified query outline will be ignored unless that mode is established using the SET FLAGS 'MODE(n)' statement or if the logical name RDMS$BIND_OUTLINE_MODE is used to define a matching mode value.

  ```
  set flags 'strategy';

  call DELETE_EMP ('Toliver', 'Alvin');
  ~S: Specified mode (99) does not match current mode − outline ignored
  ~Query Name: "TEST6"
  Tables:
    0 = EMPLOYEES2
  Conjunct: (0.LAST_NAME = <var0>) AND (0.FIRST_NAME = <var1>)
  Get     Temporary relation      Retrieval by index of relation 0:EMPLOYEES2
  ```

```
  Index name  E1_INDEX [2:2]
    Keys: (0.LAST_NAME = <var0>) AND (0.FIRST_NAME = <var1>)

set flags 'mode(99)';

call DELETE_EMP ('Toliver', 'Alvin');
~Query Name: "TEST6"
~S: Outline "(unnamed)" used
Tables:
  0 = EMPLOYEES2
Conjunct: (0.LAST_NAME = <var0>) AND (0.FIRST_NAME = <var1>)
Get     Retrieval sequentially of relation 0:EMPLOYEES2
```

- The SET FLAGS 'OUTLINE' statement can be used to have the Rdb optimizer provide a template query outline that can then be modified and incorporated into the problem query.
- Refer to the CREATE OUTLINE statement for detail of the query outline definition language.

7.1.44 RMU/DUMP/HEADER=ROW_CACHE Now Displays Whether Row Cache is Enabled

Prior to Oracle Rdb Release 7.3.1.0, the RMU RMU/DUMP/HEADER=ROW_CACHE command displayed the database−wide Row Cache definitions for an Rdb database and the RMU/DUMP/HEADER/AREA command displayed the Row Cache to be used and whether the Row Cache feature was enabled for a particular storage area. The RMU/DUMP/HEADER=ROW_CACHE command will now display whether Row Cache is enabled for ANY database storage areas. The RMU/DUMP/HEADER=ROW_CACHE command will now display the following:

```
Row caching is enabled
```

if the Row Cache feature is currently enabled for one or more database storage areas. If the Row Cache feature is not currently enabled for at least one database storage area the RMU/DUMP/HEADER=ROW_CACHE command will now display the following:

```
Row caching is disabled
```

Note that Row Cache definitions can exist in the database but a Row Cache must be enabled for a particular database storage area. The RMU/DUMP/HEADER/AREA command must still be used to see the per area Row Cache settings and whether Row Cache is currently enabled or disabled for a particular storage area.

In the following example, a database is defined with two Row Caches which are assigned to storage areas for which row caching is enabled. An RMU/DUMP/HEADER=AREA command shows the row cache parameters and whether or not row caching is enabled for each storage area. The RMU/DUMP/HEADER=ROW_CACHE command shows the database−wide Row Cache definitions. The new display "Row caching is enabled" is output since row caching is enabled for at least one storage area (in this case for all storage areas).

```
$ sql
create database filename DEVICE:[DIRECTORY]:ROW_CACHEDB
    number of cluster nodes is 1
    reserve 5 cache slots
    reserve 5 storage areas
    row cache is enabled (checkpoint updated rows to database)
    create cache tbl_phys_cache row length is 44 bytes cache size is 40 rows
    create cache idx_phys_cache row length is 432 bytes cache size is 10 rows
    create storage area tbl_sto_ar_low filename rcachedb_emp_sal_low
        cache using tbl_phys_cache
    create storage area tbl_sto_ar_high filename rcachedb_emp_sal_high
        cache using tbl_phys_cache
    create storage area idx_sto_ar filename rcachedb_emp_no
        cache using idx_phys_cache;
exit;
$ RMU/DUMP/HEADER=AREA DEVICE:[DIRECTORY]ROW_CACHEDB
*------------------------------------------------------------------------------
* Oracle Rdb V7.3-01                                  dd-mmm-yyyy hh:mm:ss.xxxx
*
* Dump of Database header
*     Database: DEVICE:[DIRECTORY]ROW_CACHEDB.RDB;1
*
*------------------------------------------------------------------------------

Database Parameters:
    Root filename is "DEVICE:[DIRECTORY]ROW_CACHEDB.RDB;1"

Storage area "RDB$SYSTEM"

    Row Caching...
      - Row caching is enabled
      - No row cache is defined for this area

Storage area "TBL_STO_AR_LOW"

    Row Caching...
      - Row caching is enabled
      - Row cache ID is 1

Storage area "TBL_STO_AR_HIGH"

    Row Caching...
      - Row caching is enabled
      - Row cache ID is 1

Storage area "IDX_STO_AR"

    Row Caching...
      - Row caching is enabled
      - Row cache ID is 2

$ RMU/DUMP/HEADER=ROW_CACHE DEVICE:[DIRECTORY]:ROW_CACHEDB
*------------------------------------------------------------------------------
* Oracle Rdb V7.3-01                                  dd-mmm-yyyy hh:mm:ss.xxxx
*
* Dump of Database header
*     Database: DEVICE:[DIRECTORY]ROW_CACHEDB.RDB;1
*
*------------------------------------------------------------------------------

Database Parameters:
    Root filename is "DEVICE:[DIRECTORY]ROW_CACHEDB.RDB;1"
    Row Caches...
      - Active row cache count is 2
      - Reserved row cache count is 5
      - Checkpoint information
          No time interval is specified
          Default source is updated rows
          Default target is database
          Default backing file directory is database directory
          RUJ Global Buffers are enabled
          No RCS sweep time interval is specified
      - WARNING: After-image journaling is disabled
      - WARNING: Fast commit is disabled

Row caching is enabled

Row cache "TBL_PHYS_CACHE"
    Cache ID number is 1
    Allocation...
      - Row slot count is 40
        - Snapshot slot count is 1000
        - Snapshots in cache disabled
      - Maximum row size allowed in cache is 44 bytes
      - Working set count is 10
      - Maximum slot reservation count is 20
      - Row replacement is enabled
    Sweeping...
      - Sweep row count is 0
      - Maximum batch I/O count is 3000
    Checkpointing...
      - Source is updated rows (database default)
      - Target is database (database default)
      - No checkpoint information available
      - Checkpoint sequence is 0
    Files...
      - Derived cache file directory is "DEVICE:[DIRECTORY]"
      - File allocation is 100 blocks
      - File extension is 100 blocks
    Hashing...
      - Hash value for logical area DBIDs is 31
      - Hash value for page numbers is 7
    Shared Memory...
      - Global Section Name is "RDM73R$1$DGA22084690010000000000001"
      - Shared memory section requirement is 16,384 bytes (1MB)

Row cache "IDX_PHYS_CACHE"
    Cache ID number is 2
    Allocation...
      - Row slot count is 10
        - Snapshot slot count is 1000
        - Snapshots in cache disabled
      - Maximum row size allowed in cache is 432 bytes
      - Working set count is 10
      - Maximum slot reservation count is 20
      - Row replacement is enabled
    Sweeping...
      - Sweep row count is 0
      - Maximum batch I/O count is 3000
    Checkpointing...
      - Source is updated rows (database default)
      - Target is database (database default)
      - No checkpoint information available
      - Checkpoint sequence is 0
    Files...
      - Derived cache file directory is "DEVICE:[DIRECTORY]"
      - File allocation is 100 blocks
      - File extension is 100 blocks
    Hashing...
      - Hash value for logical area DBIDs is 31
      - Hash value for page numbers is 7
    Shared Memory...
```

```
      − Global Section Name is "RDM73R$1$DGA22084690010000000000002"
      − Shared memory section requirement is 16,384 bytes (1MB)
```

Now an SQL statement is used to disable row caching for all storage areas. The RMU/DUMP/HEADER=AREA display shows that row caching is disabled for all storage areas and the RMU/DUMP/HEADER=ROW_CACHE command also displays "Row caching is disabled" since row caching is now disabled for all storage areas.

```
$ SQL
alter database filename DEVICE:[DIRECTORY]ROW_CACHEDB
row cache is disabled;
exit;
$ RMU/DUMP/HEADER=AREA DEVICE:[DIRECTORY]:ROW_CACHEDB
*------------------------------------------------------------------------------
* Oracle Rdb V7.3−01                                  dd−mmm−yyyy hh:mm:ss.xxxx
*
* Dump of Database header
*     Database: DEVICE:[DIRECTORY]ROW_CACHEDB.RDB;1
*
*------------------------------------------------------------------------------

Database Parameters:
    Root filename is "DEVICE:[DIRECTORY]ROW_CACHEDB.RDB;1"

Storage area "RDB$SYSTEM"

    Row Caching...
      − Row caching is disabled
      − No row cache is defined for this area

Storage area "TBL_STO_AR_LOW"

    Row Caching...
      − Row caching is disabled
      − Row cache ID is 1

Storage area "TBL_STO_AR_HIGH"
    Row Caching...
      − Row caching is disabled
      − Row cache ID is 1


Storage area "IDX_STO_AR"
    Row Caching...
      − Row caching is disabled
      − Row cache ID is 2

$ RMU/DUMP/HEADER=ROW_CACHE DEVICE:[DIRECTORY]ROW_CACHEDB
*------------------------------------------------------------------------------
* Oracle Rdb v7.3−01                                  dd−mmm−yyyy hh:mm:ss.xxxx
*
* Dump of Database header
*     Database: DEVICE:[DIRECTORY]ROW_CACHEDB.RDB;1
*
*------------------------------------------------------------------------------

Database Parameters:
    Root filename is "DEVICE:[DIRECTORY]ROW_CACHEDB.RDB;1"
    Row Caches...
      − Active row cache count is 2
      − Reserved row cache count is 5
      - Checkpoint information
          No time interval is specified
          Default source is updated rows
          Default target is database
          Default backing file directory is database directory
          RUJ Global Buffers are enabled
          No RCS sweep time interval is specified
      - WARNING: After-image journaling is disabled
      - WARNING: Fast commit is disabled

Row caching is disabled

Row cache "TBL_PHYS_CACHE"
    Cache ID number is 1
    Allocation...
      - Row slot count is 40
        - Snapshot slot count is 1000
        - Snapshots in cache disabled
      - Maximum row size allowed in cache is 44 bytes
      - Working set count is 10
      - Maximum slot reservation count is 20
      - Row replacement is enabled
    Sweeping...
      - Sweep row count is 0
      - Maximum batch I/O count is 3000
    Checkpointing...
      - Source is updated rows (database default)
      - Target is database (database default)
      - No checkpoint information available
      - Checkpoint sequence is 0
    Files...
      - Derived cache file directory is "DEVICE:[DIRECTORY]"
      - File allocation is 100 blocks
      - File extension is 100 blocks
    Hashing...
      - Hash value for logical area DBIDs is 31
      - Hash value for page numbers is 7
    Shared Memory...
      - Global Section Name is "RDM73R$1$DGA22084690010000000000001"
      - Shared memory section requirement is 16,384 bytes (1MB)

Row cache "IDX_PHYS_CACHE"
    Cache ID number is 2
    Allocation...
      - Row slot count is 10
        - Snapshot slot count is 1000
        - Snapshots in cache disabled
      - Maximum row size allowed in cache is 432 bytes
      - Working set count is 10
      - Maximum slot reservation count is 20
      - Row replacement is enabled
    Sweeping...
      - Sweep row count is 0
      - Maximum batch I/O count is 3000
    Checkpointing...
      - Source is updated rows (database default)
      - Target is database (database default)
      - No checkpoint information available
      - Checkpoint sequence is 0
    Files...
      - Derived cache file directory is "DEVICE:[DIRECTORY]"
      - File allocation is 100 blocks
      − File extension is 100 blocks
    Hashing...
      − Hash value for logical area DBIDs is 31
      − Hash value for page numbers is 7
    Shared Memory...
      − Global Section Name is "RDM73R$1$DGA22084690010000000000002"
      − Shared memory section requirement is 16,384 bytes (1MB)
$ EXIT
```

7.1.45 RMU/LOAD Now Supports CSV Formatted Files

This release of Oracle Rdb adds limited support for the CSV (comma separated list of values) format used by many tools to load data. RMU/LOAD now supports the keyword CSV, which is a variation of the DELIMITED_TEXT format currently supported by Oracle Rdb.

Usage Notes

FORMAT=CSV support is almost identical to FORMAT=DELIMITED_TEXT with some additional semantics:

- RMU/UNLOAD will create TIMESTAMP(2) format strings that are compatible with various CSV knowledgeable tools (such as Microsoft EXCEL). RMU/LOAD will implicitly convert these strings to DATE VMS during load.
- The first row is a list of column names. RMU/LOAD will implicitly skip this first row. If the CSV file is generated with multiple header lines, use the /SKIP qualifier to skip the additional lines.
- The file type defaults to .CSV

7.1.46 RMU/UNLOAD Now Supports CSV Formatted Files

This release of Oracle Rdb adds support for the CSV (comma separated list of values) format used by many tools to load data. RMU/UNLOAD now supports the keyword CSV which is a variation of the DELIMITED_TEXT format currently supported by Oracle Rdb.

Usage Notes

- Implicit conversion of DATE VMS to TIMESTAMP(2) so that formatting of text string is compatible with various CSV knowledgeable tools (such as Microsoft EXCEL).
- The first row is a list of column names. These values are formatted using the same PREFIX, SUFFIX, SEPARATOR and TERMINATOR strings as defined for the table data.
- The list of column names is re−generated when the unload file is reopened. See REOPEN_COUNT qualifier.
- The file type defaults to .CSV.

Examples

The following example shows using the RMU/UNLOAD command to generate a portable data file. The output RRD (record definition) file is suppressed using the NOFILE keyword as it is usually not useful for the target tool. The TRIM keyword is used to remove unnecessary padding spaces.

Example 7−1 Using CSV format for Microsoft EXCEL export

```
$ rmu/unload-
    /record=(nofile,format=csv,trim=trailing,term=";") -
    sql$database -
    work_status -
    ws.csv
%RMU-I-DATRECUNL,   4 data records unloaded.
$ ty ws.csv
"STATUS_CODE","STATUS_NAME","STATUS_TYPE";
"0","INACTIVE","RECORD EXPIRED";
"1","ACTIVE","FULL TIME";
"2","ACTIVE","PART TIME";
```

This example changes the delimiters for the data as required by the target loading tool.

Example 7−2 Using options to change delimiters in a CSV formatted file

```
$ rmu/unload-
    /record=(nofile,format=csv,trim=trailing,-
             term=";",pref="{",suff="}") -
    sql$database -
    current_job -
    cj.csv
...
{LAST_NAME},{FIRST_NAME},{EMPLOYEE_ID},{JOB_CODE},{DEPARTMENT_CODE},
{SUPERVISOR_ID},{JOB_START};
{Toliver},{Alvin},{00164},{DMGR},{MBMN},{00228},{1981-09-21 00:00:00.00};
...
```

7.1.47 RMU/UNLOAD Supports BITMAPPED_SCAN Optimize Option

This release of Oracle Rdb adds the keyword BITMAPPED_SCAN to the RMU/UNLOAD/Optimize qualifier.

- Bitmapped_scan
  This option requests that the Rdb optimizer attempt to perform bitmapped scan when accessing multiple indices during the unload. This option is particularly useful for RMU/UNLOAD from complex views.
  This option cannot be specified at the same time as the Sequential_Access option.
  The following shows an example of this new keyword.

```
$       define RDMS$SET_FLAGS "item_list,noprefix,strategy,detail(2)"
$       define RDMS$DEBUG_FLAGS_OUTPUT flags.log
$
$       RMU/UNLOAD-
            /OPTIMIZE=BITMAPPED_SCAN-
            RMU_UNLOAD_BITMAPPED_SCAN_4_DB-
            CURRENT_SALARY-
            CURRENT_SALARY
%RMU-I-DATRECUNL,   100 data records unloaded  5-AUG-2013 12:32:11.11.
$       SEARCH/REMAINING FLAGS.LOG "~H Request"
```

```
~H Request Information Item List: (len=11)
RDB$K_SET_REQ_OPT_PREF "0"
RDB$K_SET_REQ_OPT_BITMAPPED "1"
RDB$K_INFO_END
Tables:
  0 = SALARY_HISTORY
  1 = EMPLOYEES
Cross block of 2 entries  Q2
  Cross block entry 1
    Leaf#01 BgrOnly 0:SALARY_HISTORY Card=729        Bitmapped scan
      Bool: MISSING (0.SALARY_END)
      BgrNdx1 SH_EMPLOYEE_SS [1:1] Fan=75
        Keys: MISSING (0.SALARY_END)
  Cross block entry 2
    Get     Retrieval by index of relation 1:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [1:1]      Direct lookup
        Keys: 1.EMPLOYEE_ID = 0.EMPLOYEE_ID
$
$       deassign RDMS$DEBUG_FLAGS_OUTPUT
$       deassign RDMS$SET_FLAGS
```

7.1.48 New EDIT STRING Clause for CREATE FUNCTION and CREATE MODULE Functions

This release of Oracle Rdb adds support for the EDIT STRING clause on a function definition. It allows the value returned from the function invocation to be implicitly formatted using the EDIT STRING associated with the function. This is similar to defining an EDIT STRING on a column or domain.

The EDIT STRING clause is only used by queries in Interactive SQL.

The following example shows a simple SQL function that returns the EMPLOYEE_ID but uses the EDIT STRING for Interactive SQL to add formatting.

```
SQL> create module SAMPLE
cont>     function SHOW_EMP_ID
cont>         (in :last_name varchar(30)
cont>         ,in : birthday date)
cont>     returns integer
cont>     edit string '9999"−"9999"−"99?"VOID"'
cont>     ;
cont>     return
cont>         (select cast(employee_id as integer) * 100
cont>          from EMPLOYEES
cont>          where birthday = :birthday
cont>            and last_name = :last_name);
cont> end module;
SQL>
SQL> select SHOW_EMP_ID (last_name, birthday), last_name, first_name
cont>  from EMPLOYEES
cont>  where employee_id < '00170';
                LAST_NAME         FIRST_NAME
 0000−0164−00   Toliver           Alvin
 0000−0165−00   Smith             Terry
 0000−0166−00   Dietrich          Rick
 0000−0167−00   Kilpatrick        Janet
 0000−0168−00   Nash              Norman
 0000−0169−00   Gray              Susan
```

```
6 rows selected
SQL>
SQL> -- demonstrate the output when a NULL result is returned
SQL> select SHOW_EMP_ID ('Unknown', current_date)
cont>  from EMPLOYEES
cont>  fetch first row only;

 VOID
1 row selected
SQL>
```

The ALTER FUNCTION statement can be used to remove the edit string from a function (DROP EDIT STRING clause), or add/replace an edit string (EDIT STRING clause).

The "DROP EDIT STRING" clause removes any EDIT STRING that was previously defined for the function. No error is reported if there is no current edit string.

```
SQL> create function lib$lp_lines () returns integer;
cont> external language general
cont> general parameter style
cont> edit string 'S9(9)';
SQL> show function lib$lp_lines
Information for function LIB$LP_LINES

 Function ID is: 4
 Edit String:   S9(9)
 Language is: GENERAL
 GENERAL parameter passing style used
 Number of parameters is: 0

Parameter Name                  Data Type        Domain or Type
--------------                  ---------        --------------
                                INTEGER
        Function result datatype
        Return value is passed by value

SQL> alter function lib$lp_lines drop edit string;
```

7.1.49 Changes to RMU/VERIFY/CONSTRAINTS and ALTER TABLE Statement

With this release of Oracle Rdb, the RMU/VERIFY/CONSTRAINTS processing and the action of ALTER TABLE ... ENABLE ALL CONSTRAINTS has changed.

These changes include:

- PRIMARY KEY and UNIQUE constraints are now verified using a rewritten query that performs a single table scan. In the absence of a suitable index, the I/O required should be considerably reduced.
- NOT NULL constraints and the not null restriction for PRIMARY KEY constraints are validated using a single table scan for all such constraints on a table. This should reduce the table sequential scans in the absence of a suitable index for each constraint. Note that the side effect is that only the first failing constraint name is reported.
- When multiple tables are verified, the constraints are now ordered by table name. The goal is to make use of any buffered table rows for subsequent constraint queries. In prior versions, the constraints

were verified in (approximately) definition order which might result in other tables being read and buffered data not being available.

- During RMU/VERIFY/CONSTRAINTS or during ALTER TABLE ... ENABLE ALL CONSTRAINTS, more detailed information on the verify can be seen by defining the ITEM_LIST flag. This can be done using either the SET FLAGS statement in SQL or defining the logical name RDMS$SET_FLAGS.
  In the case of a failing NOT NULL constraint, the DBKEY of the failing row is reported. The failure status (VMS condition code) is also reported along with the associated message text.
- The algorithms used by prior releases of Oracle Rdb remain available to RMU/VERIFY/CONSTRAINTS using the new FALLBACK keyword of the /CONSTRAINTS qualifier.

```
$ DEFINE/USER RDMS$SET_FLAGS ITEM_LIST
$ RMU/VERIFY/CONSTRAINTS=FALLBACK PERSONNEL
~H Extension (VERIFY CONSTRAINTS) Item List: (len=0)
~H: ...verify constraint "COLLEGE_CODE_REQUIRED"
~H: ...verify constraint "DEPT_CODE_REQUIRED"
~H: ...verify constraint "EMPLOYEE_ID_REQUIRED"
~H: ...verify constraint "JH_EMP_ID_EXISTS"
~H: ...verify constraint "JOB_CODE_REQUIRED"
~H: ...verify constraint "SH_EMP_ID_EXISTS"
~H: 6 tables processed.
$
```

7.1.50 New SQRT Numeric Function

The function SQRT returns the square−root of the passed value expression. If the expression is NULL then the result will be NULL. Only positive values can produce a square−root. Input values are converted to DOUBLE PRECISION, if necessary. The result of the function is a DOUBLE PRECISION value.

Note

Applications which call a user defined function with the same name will continue to do so if the name is delimited (for example "SQRT") or is part of an SQL Precompiler or SQL Module Language application compiled by a prior Rdb version. In other cases, Interactive and Dynamic SQL and applications compiled using Oracle Rdb Release 7.3.1 or later will use the new built−in function.

Syntax

```
−+−> SQRT ( −+−> value_expr −+−> ) −+−−>
```

Examples

The following examples show the result of using SQRT.

Example 7−3 Example 1: Invalid request for square root of a negative value

```
SQL> select SQRT (min (salary_amount) − max (salary_amount))
cont>  from salary_history
cont>  where employee_id < '00170'
cont>  group by employee_id;
```

```
%RDB−E−ARITH_EXCEPT, truncation of a numeric value at runtime
−SYSTEM−F−FLTINV, floating invalid operation, PC=FFFFFFFF820E9362, PS=0000000B
SQL>
```

Example 7−4 Example 2: Correct query showing square root results

```
SQL> select SQRT (max (salary_amount) − min (salary_amount))
cont>  from salary_history
cont>  where employee_id < '00170'
cont>  group by employee_id;

  1.594396437527380E+002
  6.772739475278819E+001
  5.752390807307862E+001
  5.009990019950140E+001
  1.925486951396971E+002
  9.897979591815695E+001
6 rows selected
SQL>
```

7.1.51 New MOD Numeric Function

The MOD function returns the remainder of the first value expression divided by the second value expression.

If either value expression is NULL, then the result will be NULL. The result of the function is either a DOUBLE PRECISION or BIGINT value. For ANSI and ISO SQL Dialects, the result type of the function is derived from the source argument types. Any floating point argument (REAL, DOUBLE PRECISION or FLOAT) will be reflected as a DOUBLE PRECISION result. Otherwise, a BIGINT result will be returned.

If the dialect is ORACLE LEVEL1, ORACLE LEVEL2, or ORACLE LEVEL3, then Oracle semantics allow MOD to return the value of the first argument if the second evaluates to zero. Otherwise, ANSI and ISO SQL Standard behavior results in a divide by zero exception being raised.

Note

Applications which call a user defined function with the same name will continue to do so if the name is delimited (for example "MOD") or is part of an SQL Precompiler or SQL Module Language application compiled by a prior Rdb version. In other cases, Interactive and Dynamic SQL and applications compiled using Oracle Rdb Release 7.3.1 or later will use the new built−in function.

Syntax

```
−+−> MOD ( value_expr , value_expr ) −+−−>
```

Examples

The following examples show the result of using MOD. This function uses MOD in the calculation of the days in a month.

Example 7−5 Example 1: Using the MOD function

```
SQL> drop module MOD_SAMPLE if exists;
SQL> create module MOD_SAMPLE
cont>
cont>     function DAYS_IN_MONTH (in :dt date)
cont>     returns integer
cont>     comment 'Compute days in the month of the given date'
cont>     ;
cont>     begin
cont>     declare :yr constant integer = extract (year from :dt);
cont>     declare :mo constant integer = extract (month from :dt);
cont>     return case :mo
cont>             -- 30 days has September, April, June, and November
cont>             when in (4,6,9,11) then 30
cont>             when 2 then
cont>                 -- February has 28 unless it is a leap year
cont>                 case MOD(:yr, 400)
cont>                     -- leap year if divisible by 400
cont>                     when 0 then 29
cont>                     else
cont>                         case MOD(:yr, 100)
cont>                             -- not a leap year if divisible by 100
cont>                             when 0 then 28
cont>                             else
cont>                                 case MOD(:yr, 4)
cont>                                     -- leap year if divisible by 4
cont>                                     when 0 then 29
cont>                                     else 28
cont>                                 end
cont>                         end
cont>                     end
cont>             -- all the rest of 31 days
cont>             else 31
cont>            end;
cont>      end;
cont>
cont> end module;
```

7.1.52 New Data Types BINARY and BINARY VARYING

This release of Oracle Rdb adds two new data types BINARY and BINARY VARYING that allow definition of binary strings. These data types are specified by the ANSI and ISO SQL Language standard. These types would be suitable for storing small images, encrypted passwords, and so on.

Binary strings have the following characteristics:

- The SPACE octet for binary strings is X'00' (the zero valued octet). Therefore, when copying a BINARY string to a longer string it will be filled with X'00' and when comparing binary strings, the shorter string will be zero filled.
- The name VARBINARY is a synonym for BINARY VARYING.
- CONCAT (||), LIKE, MATCHING, OVERLAY, SUBSTRING, and TRIM operate on these types. The result data type of these operations will be a BINARY VARYING string.
  The clause USING { CHARACTERS | OCTETS } available for SUBSTRING and OVERLAY functions may not be used with BINARY or BINARY VARYING strings.
- POSITION, CHAR_LENGTH, and OCTET_LENGTH operate on binary string types. The

CHAR_LENGTH function is equivalent to OCTET_LENGTH for these data types.

- CONTAINING, LIKE, MATCHING and STARTING WITH operate on binary string types but all input strings must be binary strings.
- Binary string literals can be specified using the X'hex−string' literal notation.

Note

In prior releases of Oracle Rdb, such literals inherited the LITERAL CHARACTER SET but this has changed to allow binary string assignment to UNSPECIFIED.

- When declaring host language variables in C or C++, the predefined $SQL_VARBINARY should be used. This pseudo type creates a typedef in C that allows the length of the string to be passed to SQL, as frequently C zero terminated strings are inadequate to describe binary data that may need to embed X'00' values.
  The declared variable can reference the length (.len) and data (.data) as shown in the code sample below.

  ```
  $SQL_VARBINARY(65000) sql_mem;
  .
  .
  .
  sql_mem.len = MAX_STRING;
  memcpy(sql_mem.data,src_mem,sql_mem.len);
  ```

  The resulting host variable will be type compatible with BINARY and BINARY VARYING columns and variables.
- Programmers can also use the CHARACTER SET BINARY clause to provide compatible host variables.

  ```
  $SQL_VARCHAR(65000) CHARACTER SET BINARY sql_mem;
  .
  .
  .
  sql_mem.len = MAX_STRING;
  memcpy(sql_mem.data,src_mem,sql_mem.len);
  ```
- Dynamic SQL applications will see the SQLTYPE field have the type SQLDA_BINARY (913) for BINARY expressions or SQLDA_VARBINARY (909) for BINARY VARYING expressions. The symbolic names are defined in SYS$SHARE:SQL_LITERALS.H.

  ```
  #define SQLDA_VARBINARY 909
  #define SQLDA_BINARY 913
  ```

7.1.53 PERSONA SUPPORT is Enabled For All New Databases

In prior releases of Oracle Rdb, the CREATE DATABASE statement would not enable PERSONA SUPPORT by default. This meant that impersonation was done only using the OpenVMS UIC for the user. On the other hand, PERSONA SUPPORT uses the OpenVMS impersonation system services to impersonate the user including granted rights identifiers.

This release of Oracle Rdb changes the CREATE DATABASE statement to enable the PERSONA SUPPORT by default. This is shown below in this simple example.

```
SQL> create database
cont>     filename TESTING_DB;
SQL>
SQL> show database rdb$dbhandle;
Default alias:
    Oracle Rdb database in file TESTING_DB
   .
   .
   .
        Shared Memory:         Process
        Large Memory:          Disabled
        Security Checking is External (Persona support Enabled)
        System Index Compression is ENABLED
        System Index:
            Type is sorted ranked
            Prefix cardinality collection is enabled
        Logminer support is disabled
        Galaxy support is disabled
        Prestarted transactions are enabled
        Dictionary Not Required
        ACL based protections
Storage Areas in database with filename TESTING_DB
     RDB$SYSTEM                      Default and list storage area
Journals in database with filename TESTING_DB
 No Journals found
Cache Objects in database with filename TESTING_DB
 No Caches found
SQL>
```

Generally when PERSONA SUPPORT is enabled, Rdb provides much better impersonation semantics for remote database access and for services such as SQL/Services and OCI Services for Rdb. However, this new default can be disabled using the PERSONA SUPPORT IS DISABLED clause for the ALTER DATABASE or CREATE DATABASE statement.

```
SQL> alter database filename TESTING_DB
cont> security checking is external (persona support is disabled);
SQL> attach 'filename TESTING_DB';
SQL> show database rdb$dbhandle
Default alias:
    Oracle Rdb database in file TESTING_DB
        Multischema mode is disabled
   .
   .
   .
        Shared Memory:         Process
        Large Memory:          Disabled
        Security Checking is External
        System Index Compression is ENABLED
        System Index:
            Type is sorted ranked
            Prefix cardinality collection is enabled
        Logminer support is disabled
        Galaxy support is disabled
        Prestarted transactions are enabled
        Dictionary Not Required
        ACL based protections
Storage Areas in database with filename TESTING_DB
```

```
     RDB$SYSTEM                      Default and list storage area
Journals in database with filename TESTING_DB
 No Journals found
Cache Objects in database with filename TESTING_DB
 No Caches found
SQL>
```

7.1.54 New Dialects Support in SQL

This release of Oracle Rdb supports the following new dialects in SQL.

- ORACLE LEVEL3
  This includes all the behavior described for ORACLE LEVEL2 plus the following changes:
    - ♦ The same dialect rules as SQL2011 are in effect
    - ♦ The DATE data type is assumed to mean ANSI style DATE which does not include time fields
    - ♦ The CURRENT_TIMESTAMP builtin function returns a TIMESTAMP(2) type
- SQL2011
  This includes all the behavior described for SQL99 plus the following changes:
    - ♦ PRIMARY KEY or UNIQUE constraints must be evaluated at the same time or sooner than the referencing FOREIGN KEY constraints. That is, a FOREIGN KEY constraint defined as NOT DEFERRABLE may not reference a PRIMARY KEY or UNIQUE constraint defined as DEFERRABLE.

7.1.55 New WITH Clause Provides Subquery Factoring

This release of Oracle Rdb introduces the WITH clause as part of the SELECT expression. This feature, known as subquery factoring, allows the SQL programmer to simplify complex queries by creating named subqueries that can be used (possibly multiple times) in the associated SELECT expression.

The following example shows the declaration of two subquery factors, EMP and DPT, that are used in the select expression.

```
SQL> with emp as (select *
cont>              from employees inner join
cont>                   job_history using (employee_id)
cont>              where job_end is null),
cont>     dpt as (select * from departments)
cont> select e.last_name, d.department_name, m.last_name as MANAGER
cont> from emp e
cont>      left outer join dpt d using (department_code)
cont>      inner join emp m on (d.manager_id = m.employee_id)
cont> order by d.manager_id
cont> fetch first row only
cont> ;
 E.LAST_NAME      D.DEPARTMENT_NAME                MANAGER
 Siciliano        Board Manufacturing North        Toliver
1 row selected
SQL>
```

Syntax

```
select-expr =

--+--+-> select-clause ---+--+-----+
  |  |                    |  |     |
  |  +-> ( select-expr ) -+  |     |
  |  |                    |  |     |
  |  +-> TABLE table-ref -+  |     |
  |  |                    |  |     |
  |  +-> with-clause -----+  |     |
  |                          |     |
  +-- select-merge-clause <--+     |
                                   |
  +-------------- <----------------+
  |
  +--+--------------------+-+-----------------+-+--------------------+-->
     |                    | |                 | |                    |
     +-> order-by-clause -+ +-> offset-clause -+ +-> limit-to-clause -+


with-clause =

-+-> WITH -+--> subquery-name -+-----------------------------------+---+
           |                   |                                   |   |
           |                   +-> ( -+-> <name-of-column> -+-> ) -+   |
           |                          |                     |          |
           |                          +------ , <-----------+          |
           |                                                           |
           |   +-------------------------------------------------------+
           |   |
           |   +-> AS ( select-expr ) -+--> select-clause -----------------+-->
           |                                                               |
           +--------------------------- , <--------------------------------+
```

Usage Notes

- The WITH clause can appear in any place that accepts a SELECT expression. For instance, when declaring a cursor, in an INSERT ... SELECT Statement, as part of Single SELECT Statement, as part of a nested subquery, a compound statement FOR loop, and so on.
- You may reuse the subquery name in separate queries, or in more deeply nested subqueries. You may not repeat the name in the same WITH clause. See the following example.

```
SQL> with
cont>     dept as (select * from departments where department_code <> 'PRES'),
cont>     dept as (select * from jobs)
cont> select count(*), (select department_name
cont>                   from dept
cont>                   where jh.department_code = department_code)
cont>  from job_history jh, dept d
cont>  where jh.department_code = d.department_code
cont>  group by jh.department_code
cont> ;
%SQL-F-DUPVAR, Variable DEPT is already defined
SQL>
```
- If you declare a subquery factor name but do not use it, an informational message will be issued by SQL. However, the query will still be executed.

```
SQL> with
cont>     dept as (select * from departments)
cont> select count(*), (select department_name
cont>                    from departments
cont>                    where jh.department_code = department_code)
cont>  from job_history jh, departments d
cont>  where jh.department_code = d.department_code
cont>  group by jh.department_code
cont> ;
%SQL-I-VARNOTUSED, Variable "DEPT" was declared but never used

                   15   Corporate Administration
   .
   .
   .
                   12   Western U.S. Sales
26 rows selected
SQL>
```

- 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 the WITH clause specified by the ANSI and ISO SQL Database Language Standard. Therefore, to accommodate this change, Oracle Rdb has removed 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.
  The following example shows the old syntax which now produces a syntax error message.

```
SQL>
SQL> begin
cont>  with hold preserve none
 with hold preserve none
           ^
%SQL-W-LOOK_FOR_STT, Syntax error, looking for:
%SQL-W-LOOK_FOR_CON,            (, AS,
%SQL-F-LOOK_FOR_FIN,    found PRESERVE instead
```

  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;
```

Examples

The following example shows the old syntax and the new syntax for the WITH clause. The old syntax causes a syntax error now.

Example 7−6 Example 1: Using the old syntax vs the new syntax for the WITH clause

```
SQL>
SQL> begin
cont>  with hold preserve none
 with hold preserve none
           ^
%SQL-W-LOOK_FOR_STT, Syntax error, looking for:
%SQL-W-LOOK_FOR_CON,            (, AS,
%SQL-F-LOOK_FOR_FIN,    found PRESERVE instead
```

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

This example shows the use of nested subquery factoring. The nested subqueries can in turn be factored.

Example 7−7 Example 2: Using Complex Query with INSERT ... SELECT Statement

```
SQL> declare local temporary table module.EMPS
cont>     like EMPLOYEES (job_count int, sal_count int);
SQL>
SQL> insert into module.EMPS
cont>     with emp_info as
cont>         (select e.*,
cont>                 (with job_dept as
cont>                     (select jh.department_code, jh.employee_id
cont>                      from job_history jh
cont>                      where jh.employee_id = e.employee_id)
cont>                  select count(department_code) from job_dept),
cont>                 (with sal_amt as
cont>                     (select sh.salary_amount, sh.employee_id
cont>                      from salary_history sh
cont>                      where sh.employee_id = e.employee_id)
cont>                  select count(salary_amount) from sal_amt)
cont>          from employees e)
cont>     select * from emp_info
cont> ;
100 rows inserted
SQL>
```

This query finds any other employee who started a new job on a significant date for other employees.

Example 7−8 Example 3: Using subquery factoring within a UNION operator

```
SQL> select e.last_name, jh.job_start
cont> from employees e, job_history jh
cont> where e.employee_id = jh.employee_id
cont>   and jh.job_start in
cont>         (with
cont>             actual_jobs as
cont>                 (select *
cont>                  from job_history j
cont>                  where j.job_end is null)
cont>          select job_start from actual_jobs
cont>           where employee_id <> jh.employee_id
cont>          union all
cont>          with
cont>             actual_salary as
cont>                 (select *
cont>                  from salary_history s
cont>                  where s.salary_end is null)
cont>          select salary_start from actual_salary
cont>           where employee_id <> jh.employee_id)
cont> ;
 E.LAST_NAME      JH.JOB_START
 Kilpatrick       16−Aug−1980
 Nash             17-Nov-1980
 Danzig            2-Feb-1982
 Gehr              9-Sep-1981
 Clinton          28-May-1980
 Siciliano         9-Sep-1981
 Villari          16-Apr-1981
 Jackson           3-Jan-1983
 Gramby           28-May-1980
 Flynn             2-Feb-1982
 Flynn             1-Feb-1981
 Keisling          3-Jan-1983
 Klein            28-Dec-1980
 Silver            7-Aug-1982
 Belliveau        16-Apr-1981
 Crain            28-Dec-1980
 MacDonald        17-Nov-1980
17 rows selected
SQL>
```

7.1.56 DECLARE LOCAL TEMPORARY VIEW Statement

The DECLARE LOCAL TEMPORARY VIEW statement explicitly declares a local temporary view.

The metadata for a declared local temporary view is not stored in the database and cannot be shared by other modules.

This statement allows an application to define view definitions that are temporary and do not require CREATE privilege on the database.

Environment

You can use the DECLARE LOCAL TEMPORARY VIEW statement:

- In interactive SQL
- In dynamic SQL as a statement to be dynamically executed
- In a stored module as part of the module header

Format

```
DECLARE LOCAL TEMPORARY VIEW  [ alias . ] MODULE . <view-name>
```

Usage Notes

- By using a declared view, queries using those views can be simplified.
- The view definition can specify QUERY HEADER and EDIT STRING, which are only used by Interactive SQL. If the temporary view is declared in a view, then these attributes of the column are ignored.
- The view definition can specify QUERY NAME and DEFAULT VALUE FOR DTR but these attributes of the column are ignored.
- A declared local temporary view acts like a created view. Refer to the CREATE VIEW Statement for further details.

Examples

The following example declares a view which is subsequently used in a SELECT statement. The QUERY HEADER and EDIT STRING are applied by the SELECT statement.

Example 7−9 Example 1: Simplifying a query using a declared local view

```
SQL> declare local temporary view module.employee_summary
cont>  (eid
cont>      edit string 'XXBXXX'
cont>      comment is 'Employee id'
cont>  ,num_jobs
cont>      query name 'NUMBER_JOBS'
cont>  ,started
cont>      query header 'When'/'Started'
cont>  ,current_start
cont>      default value for dtr '1−Jan−1900 00:00:00.00')
cont>  as select employee_id, count(*),
cont>            min (job_start), max (job_start)
cont>        from job_history
cont>        group by employee_id;
SQL>
SQL> select * from module.employee_summary where eid <= '00164';
                          When
 EID         NUM_JOBS   Started      CURRENT_START
 00 164             2    5−JUL−1980    21−SEP−1981
1 row selected
SQL>
```

This example shows various operations on a local temporary view, including the definition of a CHECK OPTION constraint that prevents rows being inserted into the view that cannot also be retrieved by that view.

Example 7−10 Example 2: Operations on an updatable local view

```
SQL> declare local temporary view module.emp_name
cont>     (employee_id, last_name, first_name, middle_initial)
cont>     as select employee_id, last_name, first_name, middle_initial
cont>        from employees
cont>        where middle_initial is not null
cont>     with check option constraint OUT_OF_RANGE
cont> ;
SQL>
SQL> select * from module.emp_name;
 EMPLOYEE_ID   LAST_NAME        FIRST_NAME   MIDDLE_INITIAL
 00164         Toliver          Alvin        A
 00165         Smith            Terry        D
   .
   .
   .
 00435         MacDonald        Johanna      P
 00471         Herbener         James        Q
64 rows selected
SQL>
SQL> insert into module.emp_name values ('00001', 'Grey', 'Zane', NULL);
%RDB−E−INTEG_FAIL, violation of constraint OUT_OF_RANGE caused operation to fail
−RDB−F−ON_DB, on database RDB$DEFAULT_CONNECTION
SQL>
SQL> insert into module.emp_name values ('00001', 'Grey', 'Zane', 'A');
```

```
1 row inserted
SQL>
SQL> update module.emp_name
cont>  set middle_initial = 'a'
cont>  where middle_initial = 'A';
5 rows updated
SQL>
SQL> select * from module.emp_name where middle_initial = 'a';
 EMPLOYEE_ID   LAST_NAME        FIRST_NAME   MIDDLE_INITIAL
 00001         Grey             Zane         a
 00164         Toliver          Alvin        a
 00189         Lengyel          Peter        a
 00229         Robinson         Tom          a
 00416         Ames             Louie        a
5 rows selected
SQL>
SQL> rollback;
```

7.1.57 Enhancements for Buffered Read Support in SQL EXPORT DATABASE Command

This release of Oracle Rdb includes a new ROW COUNT clause as part of the EXPORT DATABASE Statement. EXPORT DATABASE now uses the buffered interface to reduce client/server exchanges while reading data rows from the source tables. In prior versions, each row was read one at a time. The default for ROW COUNT is 500 rows.

The database administrator can tune this value using the ROW COUNT clause demonstrated in the following example.

```
SQL> export database
cont>     filename MF_PERSONNEL
cont>     into SAVED_MFP
cont>     row count 1000
cont> ;
SQL>
```

7.1.58 New BITMAPPED SCAN Clauses Added to OPTIMIZE Clause

This release of Oracle Rdb allows the programmer to specify the clause OPTIMIZE FOR BITMAPPED SCAN as part of a query. This clause requests that the query optimizer attempt to use BITMAPPED SCAN if there exists multiple supporting indices in the query. The Rdb query optimizer may ignore this request if only one index is used or if no SORTED RANKED indices would be used to solve the query.

The following example shows the effect of using this new clause.

```
SQL> set flags 'strategy,detail(2)';
SQL>
SQL> select count(*)
cont>  from car
cont>  where make = 'holden'
cont>    and cyear = 1979
cont>    and colour = 'blue'
cont>    and (ctype = 'sedan' or ctype = 'wagon')
cont> optimize for bitmapped scan
cont> ;
Tables:
  0 = CAR
Aggregate: 0:COUNT (*) Q2
Leaf#01 BgrOnly 0:CAR Card=6047  Bitmapped scan
  Bool: (0.MAKE = 'holden') AND (0.CYEAR = 1979)
        AND (0.COLOUR = 'blue')
        AND ((0.CTYPE = 'sedan') OR (0.CTYPE = 'wagon'))
  BgrNdx1 IYEAR [1:1] Fan=97
    Keys: 0.CYEAR = 1979
  BgrNdx2 ICOLOUR [1:1] Fan=79
    Keys: 0.COLOUR = 'blue'
  BgrNdx3 IMAKE [1:1] Fan=79
    Keys: 0.MAKE = 'holden'
  BgrNdx4 ITYPE [(1:1)2] Fan=79
    Keys: r0: 0.CTYPE = 'wagon'
          r1: 0.CTYPE = 'sedan'

                    1
1 row selected
SQL>
```

In previous releases, the programmer would need to define the logical name RDMS$ENABLE_BITMAPPED_SCAN as 1, RDMS$SET_FLAGS as "BITMAPPED_SCAN", or use the SET FLAGS 'BITMAPPED_SCAN' statement in the application.

7.1.59 New Support for Allocations Specified Using Quantified Numeric Literal

This release of Oracle Rdb allows the database administrator to use allocation sizes using a quantified numeric literal. This shorthand notation allows the programmer to use numeric values that end in a multiplier represented by one of the following letters.

- K, meaning kilobytes
- M, meaning megabytes
- G, meaning gigabytes
- T, meaning terabytes
- P, meaning petabytes

The numeric value will be scaled according to the multiplier.

- If multiplier is K, then 1,024.
- If the multiplier is M, then 1,048,576.
- If the multiplier is G, then 1,073,741,824.
- If the multiplier is T, then 1,099,511,627,776.
- If the multiplier is P, then 1,125,899,906,842,624.

Note

Not all values specified by this notation are supported by the current release of Oracle Rdb.

These quantified numeric literals can be used with the following clauses and statements:

- ALLOCATION and SNAPSHOT ALLOCATION clause
  As part of the CREATE DATABASE Statement or IMPORT DATAABSE Statement. These clauses provide the allocation for the default storage area RDB$SYSTEM, as well as the default allocations if none are specified for specific storage areas.
- ALLOCATION and SNAPSHOT ALLOCATION clause
  As part of the CREATE STORAGE AREA clause, ADD STORAGE AREA clause, or ALTER STORAGE AREA clause. These clauses specify explicit sizes for new or altered storage area files.
- ALLOCATION clause
  As part of the CREATE, ADD or ALTER JOURNAL clause. These clauses specify explicit allocation of the new journal file.
- ALLOCATION clause
  As part of the CREATE, ADD or ALTER CACHE clause. These clauses specify explicit allocation of the caching backing file.
- MEMORY ALLOCATION clause
  As part of the GLOBAL BUFFERS clause. This value specifies the amount of virtual memory to allocate for the global buffers.

7.1.60 New SQL Functions Added

This release of Oracle Rdb adds new functions to the SYS$LIBRARY:SQL_FUNCTIONS73.SQL script. To replace the existing library of functions, first use SQL_FUNCTIONS_DROP73.SQL script and then reapply using SQL_FUNCTIONS73.SQL.

Description

- BITANDNOT (numeric−expression, numeric−expression)
  This function is used to clear bits in the first expression that are set in the second expression. First a bitwise NOT (BITNOT) is performed on the second numeric value expression and then a bitwise AND (BITAND) is performed of the first numeric value expression with the result.
  If either of the passed expressions results in NULL then the result of BITANDNOT will be NULL.
  Note that BITANDNOT is equivalent to BITAND (exp1, BITNOT (ex2)) but is more efficient.
- BITNOT (numeric−expression)
  Returns the bitwise NOT of the passed numeric value expression. If the passed expression results in NULL, then the result of BITNOT will be NULL.
- BITOR (numeric−expression, numeric−expression)
  Returns the bitwise OR of the passed numeric value expressions. If either of the passed expressions results in NULL, then the result of BITOR will be NULL.
- BITXOR (numeric−expression, numeric−expression)
  Returns the bitwise XOR of the passed numeric value expressions. If either of the passed expressions results in NULL, then the result of BITXOR will be NULL.

7.1.61 Changes and Improvements to the Rdb Optimizer and Query Compiler

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

- CAST function elimination
  In most cases, CAST actions must be executed at runtime to convert from the source data type to that specified by the CAST function. However, in some cases, the Rdb query compiler can eliminate or replace the CAST function with a literal value during query compile. This saves CPU time as the action is performed just once rather than once per row processed.
  This replacement includes the following:
  - When CAST of DATE (ANSI), DATE (VMS) or TIMESTAMP data types is performed to a compatible type of DATE or TIMESTAMP, then in many cases the CAST operator is not required.
  - CAST of string literals to DATE (ANSI), DATE (VMS), TIME, TIMESTAMP and INTERVAL can be processed at compile time. For example, CAST('2013−1−1' AS DATE ANSI) is implicitly converted to a DATE literal DATE'2013−1−1'.
  - CAST of small integer values is now done by the compiler. For example, CAST(1 AS SMALLINT) can be performed at compile time.
  - CAST of fixed length (CHAR) literal strings to varying length strings (VARCHAR) is now processed by the compiler if the character set is the same and the target VARCHAR is long enough to hold the source string, as seen in the following example:

    ```
    CAST('TABLE' AS VARCHAR(31))
    ```
- Constant Folding
  Simple arithmetic expressions involving integer or floating point literals are evaluated by the query compiler. The overall effect is smaller executable code and some reduced CPU time for queries.
  FLOAT, REAL, and DOUBLE PRECISION values are combined to produce DOUBLE PRECISION results. Integer literals (with no fractional component) are combined to produce BIGINT results.
  The side effect is that some expressions may now return DOUBLE PRECISION or BIGINT results where in prior versions they produced smaller precision results. This should not affect applications which fetch values into different data types as Oracle Rdb will perform an implicit conversion.
  This optimization includes the following:
  - Addition (+)
  - Subtraction (−)
  - Multiplication (*)
  - Division (/)
    Note that division is not performed at compile time if the divisor is a literal zero (0). Operations which are coded to explicitly divide by zero are probably expected to produce an error at runtime. Although using the SQL SIGNAL statement is now preferred, this technique has been used to terminate procedures when an incorrect input is encountered.
- Algebraic Rules
  Additive identity (zero) can be added to an expression without changing the value. The query compiler will eliminate the literal zero (0) from the expression.
  Multiply by zero will result in zero if the other operand is a not nullable expression. In this case, the expression will be replaced by zero.
  Multiplicative identity (one) can be multiplied by an expression without changing the value. The query compiler will eliminate the literal one (1) from the expression.
  The side effect is that some expressions may now return slightly different data types because the literal is no longer considered as part of the data type computation.
- Simple Predicate Elimination
  When predicates include comparison of simple expressions, then the query compiler will attempt to

eliminate them from the query predicate. For example, WHERE ('A' = 'A') will be replaced by TRUE, WHERE (2 <> 2) will be replaced with FALSE, and so on.

- Not Nullable Aware
  The query compiler is now aware of which columns have a NOT NULL NOT DEFERRABLE constraint enabled. Additionally, this attribute is also implied from any PRIMARY KEY NOT DEFERRABLE constraints.
  Using this knowledge, the query compiler can reduce (prune) the query expression. This list defines the ways in which this can occur:
    - ♦ When IS NULL is applied to a not nullable column or expression, then this predicate is replaced with FALSE.
    - ♦ When IS NOT NULL is applied to a not nullable column or expression, then this predicate is replaced with TRUE.

  The side effect is that constraints for a table are now loaded for SELECT statements.
  This optimization can be disabled using the SET FLAGS statement, or the RDMS$SET_FLAGS logical name with the value NOREWRITE(IS_NULL). The default is REWRITE(IS_NULL).
- Replace comparisons with NULL
  Queries that erroneously compare value expressions with NULL will now be replaced with a simplified UNKNOWN value. For example, a query that uses WHERE EMPLOYEE_ID = NULL will never find matching rows, because the results of the comparison (equals, not equals, greater than, less than, and so on) are always UNKNOWN.
  This optimization can be disabled using the SET FLAGS statement, or the RDMS$SET_FLAGS logical name with the value NOREWRITE(UNKNOWN). The default is REWRITE(UNKNOWN).
- Predicate Pruning
  The AND, OR and NOT operators can be simplified if the logical expressions have been reduced to TRUE, FALSE or UNKNOWN expressions. Depending on the operation, the Rdb query compiler might be able to eliminate the Boolean operator and part of the expression.
  This optimization can be disabled using the SET FLAGS statement, or the RDMS$SET_FLAGS logical name with the value NOREWRITE(BOOLEANS). The default is REWRITE(BOOLEANS).
- CASE Expression Pruning
  The prior transformation will also be applied to the Boolean WHEN expressions of a conditional expression (CASE, DECODE, NULLIF, COALESCE, NVL, NVL2, SIGN, ABS, and so on).
  In some cases, the resulting conditional expression might resolve to an equivalent conditional expression with fewer branches (some WHEN ... THEN clauses being eliminated) or a simple expression with no conditional expression (all WHEN ... THEN clauses are eliminated).
- IN Operator Simplification

  The IN operator using a subquery looks similar to the EXISTS boolean expression but it differs in its handling of NULL values. If the query compiler knows that neither source field nor the value set contain NULL, then the EXISTS expression can replace the IN operator. The EXISTS expression generates a better query solution in almost all cases.
  This optimization can be disabled using the SET FLAGS statement, or the RDMS$SET_FLAGS logical name with the value NOREWRITE(IN_CLAUSE). The default is REWRITE(IN_CLAUSE).

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

The following examples show the types of likely results.

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

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

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

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

REWRITE Flag

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

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

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

- BOOLEANS
- IN_CLAUSE
- IS_NULL
- UNKNOWN

7.1.62 Optimized NOT NULL Constraint Execution

This release of Oracle Rdb introduces a new mechanism to verify NOT NULL constraints which are executed immediately ts statement end (that is NOT DEFERRABLE). This new mechanism is more efficient (uses less code and virtual memory) than mechanisms used in prior releases. The cost of the constraint check in these cases is a fixed cost with a very small incremental cost for each extra NOT NULL constraint. The NOT NULL requirement of PRIMARY KEY constraints are also checked in the same way.

In prior releases of Oracle Rdb, each NOT NULL constraint would require its own internal query and each would be evaluated serially against the row just inserted or updated.

The following example shows an INSERT into a simple table with STRATEGY flags enabled. As can be observed, the absence of the strategy display indicates that no optimized query was used to validate these constraints.

```
SQL> set flags 'strategy,detail(2),internal,request_name';
SQL>
SQL> insert into SAMPLE
cont>     default values;
%RDB−E−INTEG_FAIL, violation of constraint SAMPLE_PK caused operation to fail
−RDB−F−ON_DB, on database RDB$DEFAULT_CONNECTION
SQL>
SQL> insert into SAMPLE (iden)
cont>     values (0);
%RDB−E−INTEG_FAIL, violation of constraint SAMPLE_DAT_NOT_NULL caused operation
to fail
−RDB−F−ON_DB, on database RDB$DEFAULT_CONNECTION
SQL>
SQL> insert into SAMPLE
cont>     values (1, 'A');
~Sn: Constraint "SAMPLE_PK" evaluated (verb)
Tables:
  0 = SAMPLE
  1 = SAMPLE
Cross block of 2 entries  Q1
  Cross block entry 1
    Conjunct: 0.DBKEY = <var0>
    Firstn: 1
    Get     Retrieval by DBK of relation 0:SAMPLE
  Cross block entry 2
    Conjunct: <agg0> <> 1
    Aggregate−F2: 0:COUNT−SINGLE (<subselect>) Q2
    Index only retrieval of relation 1:SAMPLE
      Index name  SAMPLE_NDX [1:1]
        Keys: 0.IDEN = 1.IDEN
1 row inserted
SQL>
```

Note that any DEFERRABLE constraints will be executed as in prior versions.

7.1.63 New RMU/LOAD Option CHARACTER_ENCODING_XML

When using RMU/UNLOAD/RECORD_DEFINITION/FORMAT=XML, the XML header record will, by default, use the character encoding "ISO−8859−1", as seen in the folllowing example.

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

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

This release of Oracle Rdb adds a new option, CHARACTER_ENCODING_XML, that allows the command procedure to specify an alternate character encoding. For example, if the data being unloaded is using the UTF8 character set, use this new option as shown in this example.

```
$ rmu/unload−
    /record=(nofile,format=xml,trim,character_encoding_xml="utf−8")−
    sql$database −
    employees −
    employees
%RMU−I−DATRECUNL,   100 data records unloaded  8−SEP−2013 22:21:49.54.
$
```

7.1.64 New MEMORY ALLOCATION Clause for the GLOBAL BUFFERS Definition

This release of Oracle Rdb allows the database administrator to define the size of the GLOBAL BUFFER pool using direct memory sizing as an alternate to specifying the NUMBER of pages.

- MEMORY ALLOCATION IS <mem−octets>
  The value of mem−octets is an unsigned numeric literal or a quantified numeric literal. This clause is not compatible with the NUMBER IS clause. Use just one of the keywords to define the size of the global buffers.

The following example shows the use of MEMORY ALLOCATION when creating global buffers.

```
create database
    filename TEST_DB

    allocation 110k pages
    snapshot allocation 1k pages
    global buffers are disabled (memory allocation 1m)

    create storage area AREA1
        allocation 100k pages
        snapshot allocation 2k pages
;
```

Syntax

```
global−buffer−params =

−+−> GLOBAL BUFFERS ARE −+−> ENABLED −−+−−−+
                         |             |   |
                         +−> DISABLED −+   |
                                           |
 +−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
 |
 +-+---------------------------------------------------------+-->
   |                                                         |
   +-> ( -+-+-> LARGE MEMORY IS -+-> ENABLED --+----+-+-> ) -+
          | |                    |             |    | |
          | |                    +-> DISABLED -+    | |
          | |                                       | |
          | +-> MEMORY ALLOCATION IS <mem-octets> --+ |
          | |                                       | |
          | +-> NUMBER IS <number-glo-buffers> -----+ |
          | |                                       | |
          | +-> PAGE TRANSFER VIA -+-> DISK ---+----+ |
          | |                      |           |    | |
          | |                      +-> MEMORY -+    | |
          | |                                       | |
          | +-> USER LIMIT IS <max-glo-buffers> ----+ |
          |                                           |
          +------------------ , <---------------------+
```

7.1.65 New REPLACE Statement

Bug 8929218

This release of Oracle Rdb introduces a new REPLACE statement. When a table includes a PRIMARY KEY definition, the REPLACE statement uses the key information to remove the existing matching row prior to inserting the replacement data.

The following example shows an example of the REPLACE ststement. Triggers are defined with only TRACE statements to show the order of execution during REPLACE.

```
SQL> set dialect 'sql2011';
SQL> set flags 'test_system';
SQL>
SQL> create table SAMPLE
cont>     (ident integer primary key
cont>     ,description char(40)
cont>     );
SQL>
SQL> create trigger AI_SAMPLE
cont>     after insert on SAMPLE
cont>     (trace 'after an insert')
cont>     for each row;
SQL>
SQL> create trigger BI_SAMPLE
cont>     before insert on SAMPLE
cont>     (trace 'before an insert')
cont>     for each row;
SQL>
SQL> create trigger AD_SAMPLE
cont>     after delete on SAMPLE
cont>     (trace 'after a delete')
cont>     for each row;
SQL>
SQL> create trigger BD_SAMPLE
cont>     before delete on SAMPLE
cont>     (trace 'before a delete')
cont>     for each row;
SQL>
SQL> set flags 'trace';
SQL>
SQL> -- first row
SQL> insert into SAMPLE
cont>     values (100, 'First description');
~Xt: before an insert
~Xt: after an insert
1 row inserted
SQL>
SQL> -- should fail (duplicate)
SQL> insert into SAMPLE
cont>     values (100, 'Second description');
~Xt: before an insert
~Xt: after an insert
%RDB-E-INTEG_FAIL, violation of constraint SAMPLE_PRIMARY_IDENT caused
operation to fail
-RDB-F-ON_DB, on database RDB$DEFAULT_CONNECTION
SQL>
SQL> replace into SAMPLE
cont>     values (100, 'Replace first description');
~Xt: before a delete
~Xt: after a delete
~Xt: before an insert
~Xt: after an insert
1 row replaced
SQL>
SQL> select * from SAMPLE order by ident;
       IDENT   DESCRIPTION
         100   Replace first description
1 row selected
SQL>
SQL> commit;
SQL>
```

Usage Notes

- If no PRIMARY KEY exists for the table, or it is disabled, the REPLACE statement acts exactly like an INSERT statement.
- REPLACE is a valid statement for a TRIGGER action.
- In addition to the BEFORE and AFTER INSERT triggers, REPLACE will cause BEFORE and AFTER DELETE triggers to execute.
- REPLACE is a valid compound−use statement and can be used in a stored procedure.
- It is possible that the implicit DELETE action taken by REPLACE will cause constraint execution. These constraints may prevent the DELETE actions (due to a table dependency) and therefore cause the REPLACE to fail.

Chapter 8
Documentation Corrections, Additions and Changes

This chapter provides corrections for documentation errors and omissions.

8.1 Documentation Corrections

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

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

8.1.2 Oracle Rdb Position on NFS Devices

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

Active System Files

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

Archived Data Files

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

Other Miscellaneous Files

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

Setting Up NFS

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

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

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

Read Performance Issues

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

Disclaimer

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

8.1.3 RDM$BIND_STAREA_EMERGENCY_DIR Logical Name

Bugs 19545970 and 3682207

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

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

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

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

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

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

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

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

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

The RDM$BIND_STAREA_EMERGENCY_DIR logical must:

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

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

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

8.1.4 RDMS−F−FULLAIJBKUP, Partially−Journaled Changes Made

Bug 7669735

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

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

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

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

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

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

Example 1: Initially stopping Replication on the Standby.

```
$! Stopping Replication on the Standby:

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

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

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

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

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

Example 2: Initially stopping Replication on the Master.

```
$! Stopping Replication on the Master:

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

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

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

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

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

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

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

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

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

8.1.5 Undocumented Hot Standby Logical Names

Bug 3264793

Table 8−1 Hot Standby Logical Names

Logical Name **Description**	**Default Value**	**Minimum Value**	**Maximum Value**
RDM$BIND_ALS_LOG_REOPEN_SECS Defines the number of seconds after which the ALS output file will automatically be reopened.	0 seconds (will not be reopened automatically)	0 seconds	31449600 (1 year)
RDM$BIND_ALS_LOG_REOPEN_SIZE Defines the number of blocks after which the ALS output file will automatically be reopened.	0 blocks (will not be reopened automatically)	0 blocks	Infinite
RDM$BIND_HOT_ABS_SUSPEND_SHUTDOWN Defines whether or not the AIJ backup server (ABS) should be	0	0	1

automatically suspended on graceful shutdown.			
RDM$BIND_HOT_CHECKPOINT Specifies the number of messages per server checkpoint interval. If specified, the first threshold to be exceeded (message count or elapsed time) will cause the checkpoint.	100	1	50000
RDM$BIND_HOT_CHECKPOINT_INTERVAL Specifies a checkpoint interval, in minutes, to be used in addition to the /CHECKPOINT qualifier specified at Hot Standby startup. If specified, the first threshold to be exceeded (message count or elapsed time) will cause the LRS checkpoint.	0 minutes (don't use elapsed time)	0 minutes	10080 (7 days)
RDM$BIND_HOT_IGNORE_NET_TIMEOUT Specifies whether or not to ignore network timeout parameters if the LRS process is still active.	0	0	1
RDM$BIND_HOT_LOG_REOPEN_SECS Defines the number of seconds after which the AIJSERVER output file will automatically be reopened.	0 seconds (will not be reopened automatically)	0 seconds	604800 (1 week)
RDM$BIND_HOT_LOG_REOPEN_SIZE Defines the number of blocks after which the AIJSERVER output file will automatically be reopened.	0 blocks (will not be reopened automatically)	0	Infinite
RDM$BIND_HOT_NETWORK_ALT_NODE Defines the secondary network nodename to be used in the event of primary nodename network failure. This logical name allows you to specify an alternate routing pathway to the same standby database.	None		
RDM$BIND_HOT_NETWORK_RETRY Specifies a network retry timeout interval.	120 seconds	0	1800 (30 minutes)
RDM$BIND_LCS_AIJ_SCAN_IO_COUNT Defines the number of asynchronous I/O operations to be performed simultaneously during LCS catch−up.	64	1	128
RDM$BIND_LCS_LOG_REOPEN_SECS Defines the number of seconds after which the LCS output file will automatically be reopened.	0 seconds (will not be reopened automatically)	0 seconds	31449600 (1 year)
RDM$BIND_LCS_LOG_REOPEN_SIZE Defines the number of blocks after which the LCS output file will automatically be reopened.	0 blocks (will not be reopened automatically)	0 blocks	Infinite
RDM$BIND_LCS_QUIET_TIMEOUT Defines the number of seconds to wait for the LCS process to obtain the standby database quiet−point.	600 seconds	0 seconds (wait indefinitely)	Infinite
RDM$BIND_LCS_SYNC_COMMIT_MAX Defines the number of catch−up messages to synchronize with the standby database. A message may contain multiple transactions.	128 messages	32 messages	10000 messages
RDM$BIND_LRS_LOG_REOPEN_SECS Defines the number of seconds after which the LRS output file will automatically be reopened.	0 seconds (will not be reopened automatically)	0 seconds	31449600 (1 year)

RDM$BIND_LRS_LOG_REOPEN_SIZE Defines the number of blocks after which the LRS output file will automatically be reopened.	0 blocks (will not be reopened automatically)	0 blocks	Infinite
RDM$BIND_LRS_QUIET_TIMEOUT Defines the number of seconds to wait for the LRS process to obtain the standby database quiet−point.	600	0 seconds (wait indefinitely)	Infinite
RDM$BIND_STAREA_EMERGENCY_DIR Defines an alternate device and directory for the creation of storage areas on the standby database. The logical must be defined in the LNM$SYSTEM_TABLE table and it is shared by all standby databases on that node.			

8.1.6 Clarification on Using the RMU/VERIFY SEGMENTED_STRINGS Qualifier

The /SEGMENTED_STRINGS qualifier can be used with the Oracle Rdb RMU/VERIFY command to verify all list (segmented string) data for columns in one or more database tables.

To verify segmented strings for all database tables, the command RMU/VERIFY/ALL or RMU/VERIFY/SEGMENTED_STRINGS/LAREAS or RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=* can be specified.

```
$ RMU/VERIFY/ALL PERSONNEL
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS PERSONNEL
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=* PERSONNEL
```

Neither the /SEGMENTED_STRINGS qualifier nor the /LAREAS qualifier can be specified with the /ALL qualifier since the /ALL qualifier defaults to /LAREAS=* and /SEGMENTED_STRINGS in addition to other qualifiers.

```
$ RMU/VERIFY/ALL/SEGMENTED_STRINGS/LAREAS PERSONNEL
%RMU−F−CONFLSWIT, conflicting qualifiers /ALL and /LAREAS
%RMU−F−FTL_VER, Fatal error for VERIFY operation at 16−JAN−2014 15:09:45.22
$ RMU/VERIFY/ALL/SEGMENTED_STRINGS PERSONNEL
%RMU−F−CONFLSWIT, conflicting qualifiers /ALL and /SEGMENTED_STRINGS
%RMU−F−FTL_VER, Fatal error for VERIFY operation at 16−JAN−2014 15:10:10.98
```

To verify segmented strings for individual tables, the /LAREAS qualifier must be used with the /SEGMENTED_STRING qualifier to specify one or more names of tables to be verified.

```
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=RDB$RELATIONS PERSONNEL
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=(RDB$RELATIONS,RESUMES) PERSONNEL
```

If the /SEGMENTED_STRINGS qualifier is specified, the /LAREAS qualifier must also be specified.

```
$ RMU/VERIFY/SEGMENTED_STRINGS PERSONNEL
%RMU−F−CONFLSWIT, conflicting qualifiers /SEGMENTED_STRINGS and not LAREA
%RMU−F−FTL_VER, Fatal error for VERIFY operation at 16−JAN−2014 15:11:20.16
```

When the /LAREAS qualifier is used with the /SEGMENTED_STRINGS qualifier, only logical area names that are also database table names can be specified or a fatal error will be output and the verify command will be aborted.

```
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=RESUMES MF_PERSONNEL
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=95 MF_PERSONNEL
%RMU−F−INVSEGTBL, Invalid table name 95 specified for segmented string
verification
%RMU−F−FTL_VER, Fatal error for VERIFY operation at 27−JAN−2014 15:02:39.04
$ RMU/VERIFY/SEGMENTED_STRINGS/LAREAS=SH_EMPLOYEE_ID MF_PERSONNEL
%RMU−F−INVSEGTBL, Invalid table name SH_EMPLOYEE_ID specified for segmented
string verification
%RMU−F−FTL_VER, Fatal error for VERIFY operation at 27−JAN−2014 15:03:42.15
```

If the /LOG qualifier is specified or logging is the default for the verify command, a log message will list the tables containing columns defined for segmented string data for which the segmented string data will be verified.

```
$ RMU/VERIFY/LOG/SEGMENTED_STRINGS/LAREAS=RESUMES MF_PERSONNEL
%RMU−I−BGNROOVER, beginning root verification
%RMU−I−ENDROOVER, completed root verification
%RMU−I−TBLSEGVER, Segmented strings will be verified for table RESUMES
%RMU−I−DBBOUND,   bound to database "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1"
%RMU−I−OPENAREA,  opened storage area RDB$SYSTEM for protected retrieval
%RMU−I−BGNAIPVER, beginning AIP pages verification
%RMU−I−ENDAIPVER, completed AIP pages verification
%RMU−I−BGNABMSPM, beginning ABM pages verification
%RMU−I−OPENAREA,  opened storage area MF_PERS_SEGSTR for protected retrieval
%RMU−I−ENDABMSPM, completed ABM pages verification
%RMU−I−BSGPGLARE, beginning verification of RESUMES logical area
                  as part of EMP_INFO storage area
%RMU−I−OPENAREA,  opened storage area EMP_INFO for protected retrieval
%RMU−I−ESGPGLARE, completed verification of RESUMES logical area
                  as part of EMP_INFO storage area
%RMU−I−CLOSAREAS, releasing protected retrieval lock on all storage areas
%RMU−S−ENDVERIFY, elapsed time for verification :    0 00:00:00.80
```

8.1.7 Missing Documentation for the TRANSACTION_TYPE Keyword for GET DIAGNOSTICS

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

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

The result data type is CHAR (31).

Examples

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

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

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

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

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

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

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

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

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

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

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

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

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

This is the result:

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

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

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

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

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

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

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

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

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

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

8.1.9 Corrections to the EDIT STRING Documentation

Bugs 17365476 and 17365597

- The SQL Reference Manual, Volume 1, incorrectly stated that fields following a quoted literal would have leading zeros trimmed if the literal ended with a space. This was incorrect. The trimming only takes place after a space formatting character (B).
  This is the corrected text:
  Oracle Rdb automatically trims leading zeros from the first numeric field in the output, and any numeric field following a space formatting character (B). The year (Y) and fractional seconds (*) format fields are never trimmed of leading zeros.
- To have SQL represent an OpenVMS date format without removing the leading zero from the Hour field, use the literal string for space rather than the space formatting character (B).

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

  rather than

  ```
  edit string 'YYYY-NN-DDBRR:PP:QQ.**'
  ```
- The formatting string ** represents the 100ths of a second field. Prior versions using a narrow field * would erroneously truncate the leading digits. This is corrected in this release, as the trailing digit is truncated.

8.1.10 Revised SUBSTRING Description

This release of Oracle Rdb has changed the SUBSTRING function to support binary strings as well as character strings and also provide a USING clause for specifying OCTETS or CHARACTERS units.

SUBSTRING Function (binary strings)

The SUBSTRING function returns a portion of a binary value expression. The result will be a BINARY VARYING (VARBINARY) string value. The binary data types for the source string include BINARY, BINARY VARYING or character strings with the UNSPECIFIED character set.

The FROM clause specifies the start position (position 1 is the start of the string) and the optional FOR clause specifies the string length to include in the result. The start position and string length values can be numeric value expressions.

If you specify a string length that exceeds the current length of the source string, then SQL returns only valid octets in the string and terminates the returned substring after the last valid octet.

Note that the USING clause may not be used with a binary value expression. The start position and string length always specify OCTETS.

If any operand of the SUBSTRING function is a null value, the resulting value is also null.

The following example uses the SUBSTRING function on a binary string to locate employees with invalid passwords.

Example 8−2 Using SUBSTRING (binary strings)

```
SQL> set flags 'trace';
SQL> begin
cont> for :emp
cont>      as select *
cont>   from PASSWORDS p inner join EMPLOYEES e using (employee_id)
cont>   where SUBSTRING (encrypted_password FROM 5) = x'00'
cont> do
cont>     trace :emp.last_name, 'needs password.';
cont> end for;
cont> end;
~Xt: Toliver       needs password.
~Xt: Smith         needs password.
   .
   .
   .
SQL>
```

SUBSTRING Function (character strings)

The SUBSTRING function returns a portion of a character value expression. The result will be a VARCHAR (CHARACTER VARYING) string value. The character data types for the source string can include CHAR, VARCHAR, LONG VARCHAR, as well as NATIONAL variants. The character set of the result will reflect that of the source character value expression.

The FROM clause specifies the start position (position 1 is the start of the string) and the optional FOR clause specifies the string length to include in the result. The start position and string length values can be numeric value expressions.

If you specify a string length that exceeds the current length of the source string, then SQL returns only valid characters in the string and terminates the returned substring after the last valid character.

The start position and string length specify either OCTETS or CHARACTERS within the string. The programmer can control this using one of these methods:

- Include a USING clause that specifies either the OCTETS or CHARACTERS keyword.
- Use the SET CHARACTER LENGTH statement or the CHARACTER LENGTH clause of the SQL module language header and DECLARE MODULE statement to specify whether the length value is octets or characters.
- Use the SET DIALECT or the DIALECT clause of the SQL module language header or DECLARE MODULE statement to specify a dialect. The character length will be implicitly defined by the dialect

chosen.
- The default dialect assumes OCTETS.

If any operand of the SUBSTRING function is a null value, the resulting value is also null.

The following example uses a substring in the WHERE clause of a SELECT statement. One of the SELECT statement conditions is that 4 characters starting at position 9 must equal the string 'Math'.

Example 8−3 Using SUBSTRING (character strings)

```
SQL> select * from DEGREES
cont> where SUBSTRING (degree_field from 9 for 4 using characters) = 'Math'
cont> and year_given >= 1983;
 EMPLOYEE_ID   COLLEGE_CODE   YEAR_GIVEN   DEGREE   DEGREE_FIELD
 00168         CALT                 1983   PhD      Applied Math
 00212         PRDU                 1983   MA       Applied Math
2 rows selected
SQL>
```

8.1.11 New OVERLAY Built−in Function

This release of Oracle Rdb adds support for ANSI and ISO Database Language Standard OVERLAY function. OVERLAY supports both binary strings and character strings.

Syntax

```
OVERLAY ( source−string
          PLACING replace−string
          FROM start−position
        [ FOR string−length ]
        [ USING { OCTETS | CHARACTERS } ] )
```

OVERLAY Function (binary strings)

The OVERLAY function returns a binary value expression with a portion replaced by an alternate binary string. The result will be a BINARY VARYING (VARBINARY) string value. The binary data types for the source string include BINARY, BINARY VARYING or character strings with the UNSPECIFIED character set.

The PLACING clause specifies a replacement binary value expression that will be inserted into the location specified by the FOR numeric expression. If the PLACING string evaluates to a zero length string, then the selected substring is omitted from the result. Both the source string and replacement string must be compatible with BINARY strings.

The FROM clause specifies the start position (position 1 is the start of the string) and the optional FOR clause specifies the string length to include in the source. The start position and string length values can be numeric value expressions.

If you specify a string length that exceeds the current length of the source string, then SQL returns only valid octets in the string and terminates the returned overlay after the last valid octet.

Note that the USING clause may not be used with a binary value expression. The start position and string length always specify OCTETS.

If any operand of the OVERLAY function is a null value, the resulting value is also null.

OVERLAY Function (character strings)

The OVERLAY function returns a character value expression with a portion replaced by an alternate character string. The result will be a VARCHAR (CHARACTER VARYING) string value. The character data types for the source string can include CHAR, VARCHAR, LONG VARCHAR, as well as NATIONAL variants. The character set of the result will reflect that of the source character value expression.

The PLACING clause specifies a replace character value expression that will be inserted into the location specified by the FOR numeric expression. If the PLACING string evaluates to a zero length string, then the selected substring is omitted from the result.

The FROM clause specifies the start position (position 1 is the start of the string) and the optional FOR clause specifies the string length to replace in the source. The start position and string length values can be numeric value expressions.

If you specify a string length that exceeds the current length of the source string, then SQL returns only valid characters in the string and terminates the string after the last valid character.

The start position and string length specify either OCTETS or CHARACTERS within the string. The programmer can control this using one of these methods:

- Include a USING clause that specifies either the OCTETS or CHARACTERS keyword.
- Use the SET CHARACTER LENGTH statement or the CHARACTER LENGTH clause of the SQL module language header and DECLARE MODULE statement to specify whether the length value is octets or characters.
- Use the SET DIALECT or the DIALECT clause of the SQL module language header or DECLARE MODULE statement to specify a dialect. The character length will be implicitly defined by the dialect chosen.
- The default dialect assumes OCTETS.

If any operand of the OVERLAY function is a null value, the resulting value is also null.

The following example uses an OVERLAY to create a version of the LAST_NAME column that is used to sort all Mc and Mac prefixed names together. Phonetic sorting might be done for an employee directory to make it easier to locate a name.

Example 8−4 Using OVERLAY (character string)

```
SQL> select case POSITION ('Mc' in last_name)
cont>          when 1
cont>          then OVERLAY (last_name
cont>                        placing 'Mac'
cont>                        from 1 for 2)
cont>          else last_name
cont>         end as sort_name,
cont>         last_name,
cont>         first_name
cont>  from EMPLOYEES
cont>  order by sort_name, first_name;
 SORT_NAME          LAST_NAME         FIRST_NAME
 Ames               Ames              Louie
 Andriola           Andriola          Leslie
   .
   .
   .
 Lonergan           Lonergan          Peter
 MacDonald          MacDonald         Johanna
 MacElroy           McElroy           Mary
 Manning            Manning           Kevin
 Mathias            Mathias           Susan
   .
   .
   .
```

8.1.12 Changes and Improvements to the Rdb Optimizer and Query Compiler

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

- CAST function elimination
  In most cases, CAST actions must be executed at runtime to convert from the source data type to that specified by the CAST function. However, in some cases, the Rdb query compiler can eliminate or replace the CAST function with a literal value during query compile. This saves CPU time as the action is performed just once rather than once per row processed.
  This replacement includes the following:
  - When CAST of DATE (ANSI), DATE (VMS) or TIMESTAMP data types is performed to a compatible type of DATE or TIMESTAMP, then in many cases the CAST operator is not required.
  - CAST of string literals to DATE (ANSI), DATE (VMS), TIME, TIMESTAMP and INTERVAL can be processed at compile time. For example, CAST('2013−1−1' AS DATE ANSI) is implicitly converted to a DATE literal DATE'2013−1−1'.
  - CAST of small integer values is now done by the compiler. For example, CAST(1 AS SMALLINT) can be performed at compile time.
  - CAST of fixed length (CHAR) literal strings to varying length strings (VARCHAR) is now processed by the compiler if the character set is the same and the target VARCHAR is long enough to hold the source string, as seen in the following example:

    ```
    CAST('TABLE' AS VARCHAR(31))
    ```
- Constant Folding
  Simple arithmetic expressions involving integer or floating point literals are evaluated by the query compiler. The overall effect is smaller executable code and some reduced CPU time for queries. FLOAT, REAL, and DOUBLE PRECISION values are combined to produce DOUBLE PRECISION results. Integer literals (with no fractional component) are combined to produce BIGINT results.
  The side effect is that some expressions may now return DOUBLE PRECISION or BIGINT results where in prior versions they produced smaller precision results. This should not affect applications which fetch values into different data types as Oracle Rdb will perform an implicit conversion.
  This optimization includes the following:

    - ♦ Addition (+)
    - ♦ Subtraction (−)
    - ♦ Multiplication (*)
    - ♦ Division (/)
      Note that division is not performed at compile time if the divisor is a literal zero (0). Operations which are coded to explicitly divide by zero are probably expected to produce an error at runtime. Although using the SQL SIGNAL statement is now preferred, this technique has been used to terminate procedures when an incorrect input is encountered.
- Algebraic Rules
  Additive identity (zero) can be added to an expression without changing the value. The query compiler will eliminate the literal zero (0) from the expression.
  Multiply by zero will result in zero if the other operand is a not nullable expression. In this case, the expression will be replaced by zero.
  Multiplicative identity (one) can be multiplied by an expression without changing the value. The query compiler will eliminate the literal one (1) from the expression.
  The side effect is that some expressions may now return slightly different data types because the literal is no longer considered as part of the data type computation.
- Simple Predicate Elimination
  When predicates include comparison of simple expressions, then the query compiler will attempt to eliminate them from the query predicate. For example, WHERE ('A' = 'A') will be replaced by TRUE, WHERE (2 <> 2) will be replaced with FALSE, and so on.
- Not Nullable Aware
  The query compiler is now aware of which columns have a NOT NULL NOT DEFERRABLE constraint enabled. Additionally, this attribute is also implied from any PRIMARY KEY NOT DEFERRABLE constraints.
  Using this knowledge, the query compiler can reduce (prune) the query expression. This list defines the ways in which this can occur:
    - ♦ When IS NULL is applied to a not nullable column or expression, then this predicate is replaced with FALSE.
    - ♦ When IS NOT NULL is applied to a not nullable column or expression, then this predicate is replaced with TRUE.

  The side effect is that constraints for a table are now loaded for SELECT statements.
  This optimization can be disabled using the SET FLAGS statement, or the RDMS$SET_FLAGS logical name with the value NOREWRITE(IS_NULL). The default is REWRITE(IS_NULL).
- Replace comparisons with NULL
  Queries that erroneously compare value expressions with NULL will now be replaced with a simplified UNKNOWN value. For example, a query that uses WHERE EMPLOYEE_ID = NULL will never find matching rows, because the results of the comparison (equals, not equals, greater than, less than, and so on) are always UNKNOWN.
  This optimization can be disabled using the SET FLAGS statement, or the RDMS$SET_FLAGS logical name with the value NOREWRITE(UNKNOWN). The default is REWRITE(UNKNOWN).
- Predicate Pruning
  The AND, OR and NOT operators can be simplified if the logical expressions have been reduced to TRUE, FALSE or UNKNOWN expressions. Depending on the operation, the Rdb query compiler might be able to eliminate the Boolean operator and part of the expression.
  This optimization can be disabled using the SET FLAGS statement, or the RDMS$SET_FLAGS logical name with the value NOREWRITE(BOOLEANS). The default is REWRITE(BOOLEANS).
- CASE Expression Pruning
  The prior transformation will also be applied to the Boolean WHEN expressions of a conditional expression (CASE, DECODE, NULLIF, COALESCE, NVL, NVL2, SIGN, ABS, and so on).

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

- IN Operator Simplification

  The IN operator using a subquery looks similar to the EXISTS boolean expression but it differs in its handling of NULL values. If the query compiler knows that neither source field nor the value set contain NULL, then the EXISTS expression can replace the IN operator. The EXISTS expression generates a better query solution in almost all cases.
  This optimization can be disabled using the SET FLAGS statement, or the RDMS$SET_FLAGS logical name with the value NOREWRITE(IN_CLAUSE). The default is REWRITE(IN_CLAUSE).

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

The following examples show the types of likely results.

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

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

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

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

REWRITE Flag

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

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

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

- BOOLEANS
- IN_CLAUSE
- IS_NULL
- UNKNOWN

8.1.13 Missing or Incorrect Documentation for SET AUTOMATIC TRANSLATION Command

Bug 14354801

The following errors or omissions occur in the SQL Reference Manual, Volume 4, for the SET AUTOMATIC TRANSLATION statement.

- The syntax diagram does not indicate that the *runtime−options* is optional.
- If the *runtime−options* is omitted, the default behavior is to assume 'ON' as the parameter and enable automatic translation.
- The syntax diagram does not document the Interactive SQL statements; SET NOAUTOMATIC TRANSLATION, or the alternate SET NO AUTOMATIC TRANSLATION statement.
- The arguments section asserts that DEFAULT is a legal keyword, or value for the parameter. This is incorrect; only ON or OFF are legal for the *runtime−options*.
- The following usage notes should be present.
    - ♦ The SET NOAUTOMATIC TRANSLATION and SET NO AUTOMATIC TRANSLATION statements may only be used in Interactive SQL. They are equivalent to SET AUTOMATIC TRANSLATION OFF.
    - ♦ If AUTOMATIC TRANSLATION is enabled, then translation is attempted between different versions of the table rows. For instance, after an ALTER TABLE command, where a new character set is specified for existing data. This is demonstrated in the following example.

```
SQL> create table SAMPLE (description char(20));
SQL> insert into SAMPLE (description) values ('Sample text');
1 row inserted
SQL> select description from SAMPLE;
 DESCRIPTION
 Sample text
1 row selected
SQL> alter table SAMPLE modify (description char(20) character set utf8);
SQL> select description from SAMPLE;
%RDB−E−CONVERT_ERROR, invalid or unsupported data conversion
−RDMS−E−CSETBADASSIGN, incompatible character sets prohibit the requested
assignment
SQL> set automatic translation;
SQL> select description from SAMPLE;
```

```
 DESCRIPTION
 Sample text
1 row selected
SQL>
```

Note that once the restructuring from an old version is created in the current session, it is not undone by disabling AUTOMATIC TRANSLATION.

The following examples show the usage of this statement.

Example 1: Using SET AUTOMATIC TRANSLATION command from a SQL Module Language procedure

```
procedure SET_AUTO_TRANS (sqlcode);
    SET AUTOMATIC TRANSLATION ON;
```

Or if a parameter is passed:

```
procedure SET_AUTO_TRANS
    (sqlcode,
    :on_off char(3)
    );
    SET AUTOMATIC TRANSLATION :on_off;
```

Example 2: Using SET AUTOMATIC TRANSLATION at runtime

```
SQL> declare :auto_trans char(10);
SQL> accept :auto_trans;
Enter value for AUTO_TRANS: off
SQL> set automatic translation :auto_trans;
SQL> show automatic translation;
Automatic translation: OFF
SQL>
```

8.1.14 Required Privileges for AUTHORIZATION Clause of CREATE MODULE

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

- When the AUTHORIZATION clause is used, the definer of the module is granting his/her own privileges to the specified username so that tables, columns, sequences, procedures and functions are accessed as though accessed by the definer.
  The AUTHORIZATION is expected to be the session user, or an OpenVMS rights identifier granted to that user (when SECURITY CHECKING IS EXTERNAL). If the session is run with one of the following OpenVMS privileges, then any user or rights identifier can be referenced: SYSPRV, BYPASS or IMPERSONATE.

Note

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

8.1.15 Missing Documentation for CREATE OUTLINE Statement

Bug 9864420

Prior releases of the Oracle Rdb documentation omitted a description of query outlines pertaining to views.

When Rdb compiles a query that references a view, it will implicitly use the view name to locate a matching query outline. This allows the database administrator to create partial query outlines that tune just that part of the query involving the view.

However, if the query outline is named with the same name as a view but does not follow the structure of the view then a RDMS−F−LEVEL_MISMATCH error will be reported.

The following example shows this problem.

```
SQL> create outline CURRENT_JOB
cont>   from (select * from CURRENT_JOB limit to 1 rows);
SQL>
SQL> show outline CURRENT_JOB;
     CURRENT_JOB
 Source:

-- Rdb Generated Outline :  2−SEP−2010 10:24
create outline CURRENT_JOB
id 'E9968EFAF723ED23DF59216A5DDE4C7D'
mode 0
as (
  query (
-- For loop
    subquery (
      subquery (
        EMPLOYEES 1     access path index       EMP_EMPLOYEE_ID
          join by match to
        JOB_HISTORY 0   access path index       JH_EMPLOYEE_ID
        )
      )
    )
  )
compliance optional     ;
SQL>
SQL> set flags 'strategy,detail(2)';
SQL>
SQL> select * from CURRENT_JOB limit to 1 rows;
~S: Outline "CURRENT_JOB" used
%RDMS−F−LEVEL_MISMATCH, the table/subquery nesting levels
in the query outline do not match the query
SQL>
```

To resolve this problem, the database administrator must change the name of the outline so that it is not assumed to describe the view record selection definition.

```
SQL> create outline CURRENT_JOB_REF
cont>   from (select * from CURRENT_JOB limit to 1 rows);
```

```
SQL>
SQL> set flags 'strategy,detail(2)';
SQL>
SQL> select * from CURRENT_JOB limit to 1 rows;
~S: Outline "CURRENT_JOB_REF" used
...
 LAST_NAME        FIRST_NAME   EMPLOYEE_ID   JOB_CODE   DEPARTMENT_CODE
SUPERVISOR_ID   JOB_START
 Toliver          Alvin        00164         DMGR       MBMN
00228           21-Sep-1981
1 row selected
SQL>
SQL> select * from CURRENT_JOB where employee_id = '00164'
cont> optimize using CURRENT_JOB_REF;
~S: Outline "CURRENT_JOB_REF" used
...
 LAST_NAME        FIRST_NAME   EMPLOYEE_ID   JOB_CODE   DEPARTMENT_CODE
SUPERVISOR_ID   JOB_START
 Toliver          Alvin        00164         DMGR       MBMN
00228           21-Sep-1981
1 row selected
SQL>
```

Alternatively, create the query outline on the view itself to allow it to be used more widely.

```
SQL> create outline CURRENT_JOB
cont>   on view CURRENT_JOB;
SQL>
SQL> show outline CURRENT_JOB;
     CURRENT_JOB
 Source:

-- Rdb Generated Outline :  2-SEP-2010 10:52
create outline CURRENT_JOB
-- On view CURRENT_JOB
id '9C6D98DAAF09A3E1796F7D345399028B'
mode 0
as (
  query (
-- View
    subquery (
      EMPLOYEES 1     access path index       EMP_EMPLOYEE_ID
        join by match to
      JOB_HISTORY 0   access path index       JH_EMPLOYEE_ID
      )
    )
  )
compliance optional     ;
SQL>
SQL> set flags 'strategy,detail(2)';
SQL>
SQL> select * from CURRENT_JOB limit to 1 rows;
~S: Outline "CURRENT_JOB" used
...
SQL>
```

8.1.16 Sorting Capabilities in Oracle Rdb

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

QSORT

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

QSORT is used by default if:

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

How to Alter QSORT Usage

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

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

The default value is 5000 rows.

Note

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

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

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

8.1.17 RMU /SET ROW_CACHE Command Updates

The documentation and online help for the "RMU /SET ROW_CACHE" command inadvertantly did not include the full set of allowed keywords and qualifiers.

The valid command line qualifiers for the "RMU /SET ROW_CACHE" command are:

- /ALTER − Specifies the action to take on the named cache. You must specify the cache name and at least one other option.
- /DISABLE − Disables row caching. Do not use with the Enable qualifier.
- /ENABLE − Enables row caching. Do not use with the Disable qualifier.

- /LOG − Specifies whether the processing of the command is reported to SYS$OUTPUT. Specify the Log qualifier to request log output and the Nolog qualifier to prevent it. If you specify neither, the default is the current setting of the DCL verify switch.
- /BACKING_STORE_LOCATION=devdir − Specifies the per−database default backing store location.
- /NOBACKING_STORE_LOCATION − Removes the per−database default backing store location and reverts back to the default backing store file location of the root file device and directory.

The valid values for the /ALTER qualifier are:

- NAME=cachename − Name of the cache to be modified. The cache must already be defined in the database. This is a required parameter. This parameter accepts the wildcard characters asterisk (*) and percent sign (%).
- ENABLE − Enable the cache.
- DISABLE − Disable the cache.
- DROP − Drop (delete) the cache.
- SNAPSHOT_SLOT_COUNT=n − Specify the number of snaphot slots in the cache. A value of zero disables the snapshot portion for the specified cache.
- SLOT_COUNT=n − Specify the number of slots in the cache.
- SLOT_SIZE=n − Specify the size (in bytes) of each slot in the cache.
- WORKING_SET_COUNT=n − Specify the number of working set entries for the cache. Valid values are from 1 to 100.
- BACKING_STORE_LOCATION=devdir − Specify the per−cache default backing store location.
- NOBACKING_STORE_LOCATION − Remove the per−cache default backing store location and revert back to the database default backing store file location.
- SHARED_MEMORY − Specify the shared memory type and parameters for the cache. Valid keywords are:
    - ♦ TYPE=PROCESS to specify traditional shared memory global section, which means that the database global section is located in process (P0) address space and may be paged from the process' working set as needed.
    - ♦ TYPE=RESIDENT to specify that the database global section is memory resident in process (P0) address space using OpenVMS shared page tables, which means that a system space global section is fully resident, or pinned, in memory.
    - ♦ RAD_HINT= "number" to indicate a request that memory for this shared memory should be allocated from the specified OpenVMS Alpha Resource Affinity Domain (RAD). This parameter specifies a hint to Oracle Rdb and OpenVMS about where memory should be physically allocated. It is possible that if the memory is not available, it will be allocated from other RADs in the system. For systems that do not support RADs, no RAD_HINT specification is valid.
      The RAD_HINT qualifier is only valid when the shared memory type is set to RESIDENT. Setting the shared memory type to SYSTEM or PROCESS explicitly disables any previously defined RAD hint.

      Note

      OpenVMS support for RADs is available only on the AlphaServer GS series systems. For more information about using RADs, refer to the OpenVMS Alpha Partitioning and Galaxy Guide.
    - ♦ NORAD_HINT disables the RAD hint.

The "/ALTER=(...)" qualifier may be specified multiple times on the command line. Each /ALTER qualifier specified operates on one unique cache if no wildcard character (% or *) is specified. Otherwise, each /ALTER operates on all matching cache names.

For example, the following command alters two caches:

```
$ RMU /SET ROW_CACHE MF_PERSONNEL -
        /ALTER= (   NAME = RDB$SYS_CACHE,
                    SLOT_COUNT = 800) -
        /ALTER= (   NAME = RESUMES, -
                    SLOT_SIZE=500, -
                    WORKING_SET_COUNT = 15)
```

The following command alters caches named FOOD and FOOT (and any other cache with a 4 character name with the first three characters of "FOO" defined in the database):

```
$ RMU /SET ROW_CACHE MF_PERSONNEL -
        /ALTER= (   NAME = FOO%,
                    BACKING_STORE_LOCATION=DISK$RDC:[RDC])
```

8.1.18 Documentation for the DEBUG_OPTIONS Qualifier of RMU/Unload

Bug 8447357

The RMU Help file and RMU Reference Manual are missing the description of the following qualifier for RMU/UNLOAD.

The DEBUG_OPTIONS qualifier accepts a list of keyword options.

- [NO]TRACE
  Traces the qualifier and parameter processing performed by RMU/UNLOAD. In addition, the query executed to read the table data is annotated with the TRACE statement at each COMMIT (controlled by the COMMIT_EVERY qualifier). When the logical name RDMS$SET_FLAGS is defined as "TRACE", then a line similar to the following is output after each commit is performed.

  ```
  ~Xt: 2013-04-23 15:16:16.95: Commit executed.
  ```

  The default is NOTRACE.

  ```
  $ RMU/UNLOAD/REC=(FILE=WS,FORMAT=CONTROL) SQL$DATABASE WORK_STATUS WS/DEBUG=
  TRACE
  Debug = TRACE
  * Synonyms are not enabled
  Row_Count = 500
  Message buffer: Len: 13524
  Message buffer: Sze: 27, Cnt: 500, Use: 4 Flg: 00000000
  %RMU-I-DATRECUNL,   3 data records unloaded.
  ```
- [NO]FILENAME_ONLY
  When the qualifier RECORD_DEFINITION=FORMAT:CONTROL is used, the name of the created unload file is written to the control file (.CTL). When the keyword FILENAME_ONLY is specified,

RMU/UNLOAD will prune the output file specification to show only the file name and type. The default is NOFILENAME_ONLY.

```
$ RMU/UNLOAD/REC=(FILE=TT:,FORMAT=CONTROL) SQL$DATABASE WORK_STATUS WS/DEBUG=
FILENAME
--
-- SQL*Loader Control File
--   Generated by: RMU/UNLOAD
--   Version:      Oracle Rdb X7.2-00
--   On:           23-APR-2009 11:12:46.29
--
LOAD DATA
INFILE 'WS.UNL'
APPEND
INTO TABLE "WORK_STATUS"
(
 STATUS_CODE                     POSITION(1:1) CHAR NULLIF (RDB$UL_NB1 = '1')
,STATUS_NAME                     POSITION(2:9) CHAR NULLIF (RDB$UL_NB2 = '1')
,STATUS_TYPE                     POSITION(10:23) CHAR NULLIF (RDB$UL_NB3 = '1')
-- NULL indicators
,RDB$UL_NB1           FILLER POSITION(24:24) CHAR -- indicator for STATUS_CODE
,RDB$UL_NB2           FILLER POSITION(25:25) CHAR -- indicator for STATUS_NAME
,RDB$UL_NB3           FILLER POSITION(26:26) CHAR -- indicator for STATUS_TYPE
)
%RMU-I-DATRECUNL,   3 data records unloaded.
```

- [NO]HEADER
  This keyword controls the output of the header in the control file. To suppress the header, use NOHEADER. The default is HEADER.
- APPEND, INSERT, REPLACE, TRUNCATE
  These keywords control the text that is output prior to the INTO TABLE clause in the control file. The default is APPEND and only one of these options can be specified.

8.1.19 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>
```

8.1.20 RMU /VERIFY Process Quotas and Limits Clarification

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

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

8.1.21 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.

8.1.22 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 V7.3 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;
```

8.1.23 RDM$BIND_MAX_DBR_COUNT Documentation Clarification

Bugs 1495227 and 3916606

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

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

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

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

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

Per−Database Value

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

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

8.1.24 Database Server Process Priority Clarification

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

Individual server priorities can be explicitly controlled via system−wide logical names as described in Table 8−2.

Table 8−2 Server Process Priority Logical Names

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

The RDMAIJSERVER account for Hot Standby is created specifying an account priority of 15. The priority of AIJ server processes on your system can be restricted with the system−wide logical name

RDM$BIND_AIJSRV_PRIORITY. If this logical name is defined to a value less than 15, an AIJ server process will adjust its base priority to the value specified when the AIJ server process starts. Values from 0 to 31 are allowed for RDM$BIND_AIJSRV_PRIORITY, but the process is not able to raise its priority above the RDMAIJSERVER account value.

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

8.1.25 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.

8.1.26 RDM$BIND_LOCK_TIMEOUT_INTERVAL Overrides the Database Parameter

Bug 2203700

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

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

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

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

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

8.1.27 Missing Tables Descriptions for the RDBEXPERT Collection Class

Appendix B in the Oracle Rdb7 Guide to Database Performance and Tuning describes the event−based data tables in the formatted database for the Oracle Rdb PERFORMANCE and RDBEXPERT collection classes. This section describes the missing tables for the RDBEXPERT collection class.

Table 8−3 shows the TRANS_TPB table.

Table 8−3 Columns for Table EPC$1_221_TRANS_TPB

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_POINT	DATE VMS	
CLIENT_PC	INTEGER	
STREAM_ID	INTEGER	
TRANS_ID	VARCHAR(16)	
TRANS_ID_STR_ID	INTEGER	STR_ID_DOMAIN
TPB	VARCHAR(127)	
TPB_STR_ID	INTEGER	STR_ID_DOMAIN

Table 8−4 shows the TRANS_TPB_ST table. An index is provided for this table. It is defined with column STR_ID, duplicates are allowed, and the type is sorted.

Table 8−4 Columns for Table EPC$1_221_TRANS_TPB_ST

Column Name	Data Type	Domain
STR_ID	INTEGER	STR_ID_DOMAIN
SEGMENT_NUMBER	SMALLINT	SEGMENT_NUMBER_DOMAIN
STR_SEGMENT	VARCHAR(128)	

8.1.28 Missing Columns Descriptions for Tables in the Formatted Database

Some of the columns were missing from the tables in Appendix B in the Oracle Rdb7 Guide to Database Performance and Tuning. The complete table definitions are described in this section.

Table 8−5 shows the DATABASE table.

Table 8−5 Columns for Table EPC$1_221_DATABASE

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_POINT	DATE VMS	
CLIENT_PC	INTEGER	
STREAM_ID	INTEGER	
DB_NAME	VARCHAR(255)	
DB_NAME_STR_ID	INTEGER	STR_ID_DOMAIN
IMAGE_FILE_NAME	VARCHAR(255)	
IMAGE_FILE_NAME_STR_ID	INTEGER	STR_ID_DOMAIN

Table 8−6 shows the REQUEST_ACTUAL table.

Table 8−6 Columns for Table EPC$1_221_REQUEST_ACTUAL

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_START	DATE VMS	
TIMESTAMP_END	DATE VMS	
DBS_READS_START	INTEGER	
DBS_WRITES_START	INTEGER	
RUJ_READS_START	INTEGER	
RUJ_WRITES_START	INTEGER	
AIJ_WRITES_START	INTEGER	
ROOT_READS_START	INTEGER	
ROOT_WRITES_START	INTEGER	
BUFFER_READS_START	INTEGER	
GET_VM_BYTES_START	INTEGER	
FREE_VM_BYTES_START	INTEGER	
LOCK_REQS_START	INTEGER	
REQ_NOT_QUEUED_START	INTEGER	
REQ_STALLS_START	INTEGER	
REQ_DEADLOCKS_START	INTEGER	
PROM_DEADLOCKS_START	INTEGER	
LOCK_RELS_START	INTEGER	

LOCK_STALL_TIME_START	INTEGER
D_FETCH_RET_START	INTEGER
D_FETCH_UPD_START	INTEGER
D_LB_ALLOK_START	INTEGER
D_LB_GBNEEDLOCK_START	INTEGER
D_LB_NEEDLOCK_START	INTEGER
D_LB_OLDVER_START	INTEGER
D_GB_NEEDLOCK_START	INTEGER
D_GB_OLDVER_START	INTEGER
D_NOTFOUND_IO_START	INTEGER
D_NOTFOUND_SYN_START	INTEGER
S_FETCH_RET_START	INTEGER
S_FETCH_UPD_START	INTEGER
S_LB_ALLOK_START	INTEGER
S_LB_GBNEEDLOCK_START	INTEGER
S_LB_NEEDLOCK_START	INTEGER
S_LB_OLDVER_START	INTEGER
S_GB_NEEDLOCK_START	INTEGER
S_GB_OLDVER_START	INTEGER
S_NOTFOUND_IO_START	INTEGER
S_NOTFOUND_SYN_START	INTEGER
D_ASYNC_FETCH_START	INTEGER
S_ASYNC_FETCH_START	INTEGER
D_ASYNC_READIO_START	INTEGER
S_ASYNC_READIO_START	INTEGER
AS_READ_STALL_START	INTEGER
AS_BATCH_WRITE_START	INTEGER
AS_WRITE_STALL_START	INTEGER
BIO_START	INTEGER
DIO_START	INTEGER
PAGEFAULTS_START	INTEGER
PAGEFAULT_IO_START	INTEGER
CPU_START	INTEGER
CURRENT_PRIO_START	SMALLINT
VIRTUAL_SIZE_START	INTEGER
WS_SIZE_START	INTEGER
WS_PRIVATE_START	INTEGER
WS_GLOBAL_START	INTEGER
CLIENT_PC_END	INTEGER
STREAM_ID_END	INTEGER
REQ_ID_END	INTEGER
COMP_STATUS_END	INTEGER

REQUEST_OPER_END	INTEGER	
TRANS_ID_END	VARCHAR(16)	
TRANS_ID_END_STR_ID	INTEGER	STR_ID_DOMAIN
DBS_READS_END	INTEGER	
DBS_WRITES_END	INTEGER	
RUJ_READS_END	INTEGER	
RUJ_WRITES_END	INTEGER	
AIJ_WRITES_END	INTEGER	
ROOT_READS_END	INTEGER	
ROOT_WRITES_END	INTEGER	
BUFFER_READS_END	INTEGER	
GET_VM_BYTES_END	INTEGER	
FREE_VM_BYTES_END	INTEGER	
LOCK_REQS_END	INTEGER	
REQ_NOT_QUEUED_END	INTEGER	
REQ_STALLS_END	INTEGER	
REQ_DEADLOCKS_END	INTEGER	
PROM_DEADLOCKS_END	INTEGER	
LOCK_RELS_END	INTEGER	
LOCK_STALL_TIME_END	INTEGER	
D_FETCH_RET_END	INTEGER	
D_FETCH_UPD_END	INTEGER	
D_LB_ALLOK_END	INTEGER	
D_LB_GBNEEDLOCK_END	INTEGER	
D_LB_NEEDLOCK_END	INTEGER	
D_LB_OLDVER_END	INTEGER	
D_GB_NEEDLOCK_END	INTEGER	
D_GB_OLDVER_END	INTEGER	
D_NOTFOUND_IO_END	INTEGER	
D_NOTFOUND_SYN_END	INTEGER	
S_FETCH_RET_END	INTEGER	
S_FETCH_UPD_END	INTEGER	
S_LB_ALLOK_END	INTEGER	
S_LB_GBNEEDLOCK_END	INTEGER	
S_LB_NEEDLOCK_END	INTEGER	
S_LB_OLDVER_END	INTEGER	
S_GB_NEEDLOCK_END	INTEGER	
S_GB_OLDVER_END	INTEGER	
S_NOTFOUND_IO_END	INTEGER	
S_NOTFOUND_SYN_END	INTEGER	
D_ASYNC_FETCH_END	INTEGER	
S_ASYNC_FETCH_END	INTEGER	

D_ASYNC_READIO_END	INTEGER
S_ASYNC_READIO_END	INTEGER
AS_READ_STALL_END	INTEGER
AS_BATCH_WRITE_END	INTEGER
AS_WRITE_STALL_END	INTEGER
BIO_END	INTEGER
DIO_END	INTEGER
PAGEFAULTS_END	INTEGER
PAGEFAULT_IO_END	INTEGER
CPU_END	INTEGER
CURRENT_PRIO_END	SMALLINT
VIRTUAL_SIZE_END	INTEGER
WS_SIZE_END	INTEGER
WS_PRIVATE_END	INTEGER
WS_GLOBAL_END	INTEGER

Table 8−7 shows the TRANSACTION table.

Table 8−7 Columns for Table EPC$1_221_TRANSACTION

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_START	DATE VMS	
TIMESTAMP_END	DATE VMS	
CLIENT_PC_START	INTEGER	
STREAM_ID_START	INTEGER	
LOCK_MODE_START	INTEGER	
TRANS_ID_START	VARCHAR(16)	
TRANS_ID_START_STR_ID	INTEGER	STR_ID_DOMAIN
GLOBAL_TID_START	VARCHAR(16)	
GLOBAL_TID_START_STR_ID	INTEGER	STR_ID_DOMAIN
DBS_READS_START	INTEGER	
DBS_WRITES_START	INTEGER	
RUJ_READS_START	INTEGER	
RUJ_WRITES_START	INTEGER	
AIJ_WRITES_START	INTEGER	
ROOT_READS_START	INTEGER	
ROOT_WRITES_START	INTEGER	
BUFFER_READS_START	INTEGER	
GET_VM_BYTES_START	INTEGER	

FREE_VM_BYTES_START	INTEGER	
LOCK_REQS_START	INTEGER	
REQ_NOT_QUEUED_START	INTEGER	
REQ_STALLS_START	INTEGER	
REQ_DEADLOCKS_START	INTEGER	
PROM_DEADLOCKS_START	INTEGER	
LOCK_RELS_START	INTEGER	
LOCK_STALL_TIME_START	INTEGER	
D_FETCH_RET_START	INTEGER	
D_FETCH_UPD_START	INTEGER	
D_LB_ALLOK_START	INTEGER	
D_LB_GBNEEDLOCK_START	INTEGER	
D_LB_NEEDLOCK_START	INTEGER	
D_LB_OLDVER_START	INTEGER	
D_GB_NEEDLOCK_START	INTEGER	
D_GB_OLDVER_START	INTEGER	
D_NOTFOUND_IO_START	INTEGER	
D_NOTFOUND_SYN_START	INTEGER	
S_FETCH_RET_START	INTEGER	
S_FETCH_UPD_START	INTEGER	
S_LB_ALLOK_START	INTEGER	
S_LB_GBNEEDLOCK_START	INTEGER	
S_LB_NEEDLOCK_START	INTEGER	
S_LB_OLDVER_START	INTEGER	
S_GB_NEEDLOCK_START	INTEGER	
S_GB_OLDVER_START	INTEGER	
S_NOTFOUND_IO_START	INTEGER	
S_NOTFOUND_SYN_START	INTEGER	
D_ASYNC_FETCH_START	INTEGER	
S_ASYNC_FETCH_START	INTEGER	
D_ASYNC_READIO_START	INTEGER	
S_ASYNC_READIO_START	INTEGER	
AS_READ_STALL_START	INTEGER	
AS_BATCH_WRITE_START	INTEGER	
AS_WRITE_STALL_START	INTEGER	
AREA_ITEMS_START	VARCHAR(128)	
AREA_ITEMS_START_STR_ID	INTEGER	STR_ID_DOMAIN
BIO_START	INTEGER	
DIO_START	INTEGER	
PAGEFAULTS_START	INTEGER	
PAGEFAULT_IO_START	INTEGER	
CPU_START	INTEGER	

CURRENT_PRIO_START	SMALLINT
VIRTUAL_SIZE_START	INTEGER
WS_SIZE_START	INTEGER
WS_PRIVATE_START	INTEGER
WS_GLOBAL_START	INTEGER
CROSS_FAC_2_START	INTEGER
CROSS_FAC_3_START	INTEGER
CROSS_FAC_7_START	INTEGER
CROSS_FAC_14_START	INTEGER
DBS_READS_END	INTEGER
DBS_WRITES_END	INTEGER
RUJ_READS_END	INTEGER
RUJ_WRITES_END	INTEGER
AIJ_WRITES_END	INTEGER
ROOT_READS_END	INTEGER
ROOT_WRITES_END	INTEGER
BUFFER_READS_END	INTEGER
GET_VM_BYTES_END	INTEGER
FREE_VM_BYTES_END	INTEGER
LOCK_REQS_END	INTEGER
REQ_NOT_QUEUED_END	INTEGER
REQ_STALLS_END	INTEGER
REQ_DEADLOCKS_END	INTEGER
PROM_DEADLOCKS_END	INTEGER
LOCK_RELS_END	INTEGER
LOCK_STALL_TIME_END	INTEGER
D_FETCH_RET_END	INTEGER
D_FETCH_UPD_END	INTEGER
D_LB_ALLOK_END	INTEGER
D_LB_GBNEEDLOCK_END	INTEGER
D_LB_NEEDLOCK_END	INTEGER
D_LB_OLDVER_END	INTEGER
D_GB_NEEDLOCK_END	INTEGER
D_GB_OLDVER_END	INTEGER
D_NOTFOUND_IO_END	INTEGER
D_NOTFOUND_SYN_END	INTEGER
S_FETCH_RET_END	INTEGER
S_FETCH_UPD_END	INTEGER
S_LB_ALLOK_END	INTEGER
S_LB_GBNEEDLOCK_END	INTEGER
S_LB_NEEDLOCK_END	INTEGER
S_LB_OLDVER_END	INTEGER

S_GB_NEEDLOCK_END	INTEGER	
S_GB_OLDVER_END	INTEGER	
S_NOTFOUND_IO_END	INTEGER	
S_NOTFOUND_SYN_END	INTEGER	
D_ASYNC_FETCH_END	INTEGER	
S_ASYNC_FETCH_END	INTEGER	
D_ASYNC_READIO_END	INTEGER	
S_ASYNC_READIO_END	INTEGER	
AS_READ_STALL_END	INTEGER	
AS_BATCH_WRITE_END	INTEGER	
AS_WRITE_STALL_END	INTEGER	
AREA_ITEMS_END	VARCHAR(128)	
AREA_ITEMS_END_STR_ID	INTEGER	STR_ID_DOMAIN
BIO_END	INTEGER	
DIO_END	INTEGER	
PAGEFAULTS_END	INTEGER	
PAGEFAULT_IO_END	INTEGER	
CPU_END	INTEGER	
CURRENT_PRIO_END	SMALLINT	
VIRTUAL_SIZE_END	INTEGER	
WS_SIZE_END	INTEGER	
WS_PRIVATE_END	INTEGER	
WS_GLOBAL_END	INTEGER	
CROSS_FAC_2_END	INTEGER	
CROSS_FAC_3_END	INTEGER	
CROSS_FAC_7_END	INTEGER	
CROSS_FAC_14_END	INTEGER	

Table 8−8 shows the REQUEST_BLR table.

Table 8−8 Columns for Table EPC$1_221_REQUEST_BLR

Column Name	Data Type	Domain
COLLECTION_RECORD_ID	SMALLINT	COLLECTION_RECORD_ID_DOMAIN
IMAGE_RECORD_ID	INTEGER	IMAGE_RECORD_ID_DOMAIN
CONTEXT_NUMBER	INTEGER	CONTEXT_NUMBER_DOMAIN
TIMESTAMP_POINT	DATE VMS	
CLIENT_PC	INTEGER	
STREAM_ID	INTEGER	
REQ_ID	INTEGER	
TRANS_ID	VARCHAR(16)	
TRANS_ID_STR_ID	INTEGER	STR_ID_DOMAIN

REQUEST_NAME	VARCHAR(31)	
REQUEST_NAME_STR_ID	INTEGER	STR_ID_DOMAIN
REQUEST_TYPE	INTEGER	
BLR	VARCHAR(127)	
BLR_STR_ID	INTEGER	STR_ID_DOMAIN

8.2 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
```

8.3 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 the Oracle Rdb Documentation web page:

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

Customers should contact their Oracle representative to purchase printed documentation.

Chapter 9
Known Problems and Restrictions

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

9.1 Known Problems and Restrictions in All Interfaces

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

9.1.1 Known Problems With REVERSE Indices

Bugs 18496095 and 18496853

Oracle Rdb Release 7.3.1 introduced a new REVERSE attribute for use with SORTED and SORTED RANKED indices. Unfortunately, this functionality has had several reported issues for both Interity and Alpha platforms. Therefore, this functionality has been disabled in this release of Oracle Rdb (7.3.1.2).

Oracle Rdb is continuing to improve and test this functionality and plans to re−enable this functionality in a future release of Oracle Rdb.

If you currently have indices defined with the REVERSE attribute, they will automatically be considered as MAINTENANCE IS DISABLED and will not be updated by DELETE, INSERT or UPDATE statements, nor used by the query optimizer. At your earliest convenience, they should be dropped and re−created without the REVERSE clause.

Oracle apologizes for the inconvenience and removal of this functionality.

9.1.2 Null Elimination Warning Not Generated for Some Aggregates

The current release of Oracle Rdb does not correctly return the null elimination warning for aggregates that are optimized to use special index operations against SORTED indices, even when the dialect requests this returned warning.

The null elimination warning is returned when using one of the following SQL dialects: SQL92, SQL99, ORACLE LEVEL1, or ORACLE LEVEL2.

The list of affected aggregate functions are:

- MAX (column)
- MAX (DISTINCT column)
- MIN (column)
- MIN (DISTINCT column)

When these functions are encountered and the optimizer is able to apply "Max key lookup" or "Min key lookup" then Rdb currently disables the return of null elimination warning in preference to improved query performance.

In addition, COUNT(column) and COUNT(DISTINCT column) do not correctly return the null elimination warning against SORTED RANKED indices.

This restriction does not affect COUNT(*) which correctly reports the null elimination warning when "Index count lookup" or "Index distinct lookup" is used.

9.1.3 RMU/BACKUP/AFTER_JOURNAL Ignores the Default After Journal Compression Setting

The Oracle Rdb RMU/BACKUP/AFTER_JOURNAL command ignores the default compression setting to be used for backing up Rdb database After Journal files. The default After Journal file compression setting can be set in the Rdb database root file using the /BACKUP qualifier of the RMU/SET AFTER_JOURNAL command. This problem has existed since the Oracle Rdb V7.3−1 release.

The compression setting for the backup of an After Journal file can be set by the /COMPRESSION qualifier of the RMU/BACKUP/AFTER_JOURNAL command. But, due to this problem, if the /COMPRESSION qualifier is not specified with the RMU/BACKUP/AFTER_JOURNAL command, the default After Journal file backup compression setting, if it is set in the database root, is ignored and no compression is used for the After Journal file backup file.

The following example shows this problem. A default compression setting to be used when backing up database After Journal files is saved in the database root by the RMU/SET AFTER_JOURNAL command. But, when the RMU/BACKUP/AFTER_JOURNAL command to backup After Journal files is executed without specifying a compression setting for the backup, the default compression setting specified in the database root is ignored, and the backup command does not compress the After Journal backup file.

```
$ rmu/set after_journal mf_personnel/backup=compression=ZLIB:5/log
%RMU−I−LOGMODVAL,     modified AIJ backup compression level to 5
$ rmu/backup/after_journal/log mf_personnel mfp.aij_bck
%RMU−I−AIJBCKBEG, beginning after−image journal backup operation
%RMU−I−OPERNOTIFY, system operator notification: AIJ backup operation started
%RMU−I−AIJBCKSEQ, backing up after−image journal sequence number 0
%RMU−I−LOGBCKAIJ, backing up after−image journal RDB$JOURNAL at 13:50:15.35
%RMU−I−LOGCREBCK, created backup file DEVICE:[DIRECTORY]MFP.AIJ_BCK;10
%RMU−I−QUIETPT, waiting for database quiet point at 24−SEP−2014 13:50:15.38
%RMU−I−LOGAIJBCK, backed up 1 committed transaction at 13:50:15.38
%RMU−I−AIJMODSEQ, next AIJ file sequence number will be 1
%RMU−I−OPERNOTIFY, system operator notification: AIJ backup operation completed
%RMU−I−AIJBCKEND, after−image journal backup operation completed successfully
%RMU−I−LOGAIJJRN, backed up 1 after−image journal at 13:50:15.38
%RMU−I−LOGAIJBLK, backed up 3 after−image journal blocks at 13:50:15.38
%RMU−I−LOGAIJBCK, backed up 1 committed transaction at 13:50:15.38
```

To avoid this problem, specify the desired compression setting by using the /COMPRESSION qualifier with the RMU/BACKUP/AFTER_JOURNAL command.

```
$ rmu/backup/after_journal/COMPRESSION=ZLIB:4/log mf_personnel mfp.aij_bck
%RMU−I−AIJBCKBEG, beginning after−image journal backup operation
%RMU−I−OPERNOTIFY, system operator notification: AIJ backup operation started
%RMU−I−AIJBCKSEQ, backing up after−image journal sequence number 0
%RMU−I−LOGBCKAIJ, backing up after−image journal RDB$JOURNAL at 13:54:27.27
%RMU−I−LOGCREBCK, created backup file DEVICE:[DIRECTORY]MFP.AIJ_BCK;11
%RMU−I−QUIETPT, waiting for database quiet point at 24−SEP−2014 13:54:27.32
%RMU−I−LOGAIJBCK, backed up 1 committed transaction at 13:54:27.32
%RMU−I−AIJMODSEQ, next AIJ file sequence number will be 1
%RMU−I−OPERNOTIFY, system operator notification: AIJ backup operation completed
%RMU−I−AIJBCKEND, after−image journal backup operation completed successfully
%RMU−I−LOGAIJJRN, backed up 1 after−image journal at 13:54:27.32
```

```
%RMU−I−LOGAIJBLK, backed up 3 after−image journal blocks at 13:54:27.32
%RMU−I−LOGAIJBCK, backed up 1 committed transaction at 13:54:27.32
```

This problem will be corrected in a future release of Oracle Rdb V7.3. RMU Backup After_Journal will make use of the default established by the RMU/SET AFTER_JOURNAL/BACKUPS=COMPRESSION attribute.

9.1.4 RMU /VERIFY /KEY_VALUES May Fail on Some Indices

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

The following example shows the reported error.

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

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

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

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

The following example shows the problem.

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

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

9.1.6 Possible Incorrect Results When Using Partitioned Descending Indexes

Bug 6129797

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

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

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

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

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

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

create index x4 on mesa (id asc , m4 asc) ;

! The following index contains ascending segments followed by descending
! segments and thus causes the query to return the wrong result.
!
! Note that the query works if both segments are either ascending or descending.
!
create index y4 on rasa (id asc , r4 desc)
       store using (id, r4)
       in fooa with limit of (1, 'g' )
       otherwise in foob ;
commit;

! Problem #1:
!
! the following query returns correctly 3 rows on Alpha but 1 row on IA64:

SQL> select m.id, m.m4, r.r4 from
   mesa m inner join rasa r on (m.id = r.id);
           1   m      m
           1   m      k
           1   m      e
3 rows selected

SQL> select m.id, m.m4, r.r4 from mesa m inner join rasa r on (m.id = r.id);
           1   m      e
1 row selected
```

```
! Problem #2:
!
! The following query using reverse scan returns 2 rows incorrectly on Alpha
! but 3 rows correctly on IA64:
!

SQL> select id, r4 from rasa where id = 1 and r4 <= 'm' order by id, r4;
Tables:
  0 = RASA
Index only retrieval of relation 0:RASA
  Index name  Y4 [2:1]   Reverse Scan
    Keys: (0.ID = 1) AND (0.R4 <= 'm')
        ID   R4
         1   k
         1   m
2 rows selected

SQL> select id, r4 from rasa where id = 1 and r4 <= 'm' order by id, r4;
Tables:
  0 = RASA
Index only retrieval of relation 0:RASA
  Index name  Y4 [2:1]   Reverse Scan
    Keys: (0.ID = 1) AND (0.R4 <= 'm')
        ID   R4
         1   e
         1   k
         1   m
3 rows selected
```

This problem is related to the construction and comparison of the descending key values while processing the index partitions.

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

9.1.7 Remote Attach Stalls Before Detecting a Node is Unreachable

Bug 7681548

A remote attach can stall for a noticeable period, typically 75 seconds, before detecting a node is unreachable.

The following example shows the expected error message when attempting to access a database on a node that is not reachable. The problem is that when the value of the parameter SQL_NETWORK_TRANSPORT_TYPE in the file RDB$CLIENT_DEFAULTS.DAT is not specifically set to DECNET (in UPPER CASE), a stall of typically 75 seconds will happen before you get the expected error message.

```
SQL> attach 'file 1::disk1:[dbdir]db';
%SQL−F−ERRATTDEC, Error attaching to database 1::disk1:[dbdir]db
−RDB−F−IO_ERROR, input or output error
−SYSTEM−F−UNREACHABLE, remote node is not currently reachable
```

There are two possible ways to avoid the stall and get the error message after a user configurable period of time or instantly: decrease the value of the TCPIP parameter TCP_KEEPINIT, or explicitly specify SQL_NETWORK_TRANSPORT_TYPE as DECNET (in UPPER CASE).

- The default behavior when attempting to connect to an unreachable node via TCPIP is to stall 75 seconds before returning an error. The stall time is configurable in TCPIP via the parameter TCP_KEEPINIT which is expressed in units of 500 ms. The default value of TCP_KEEPINIT is 150 which corresponds to a 75 second stall.
- When connecting via DECnet, the error message is typically returned instantly so a significant stall will not be seen in this case. However, the value of the parameter SQL_NETWORK_TRANSPORT_TYPE is case sensitive so for DECnet to be selected as the transport, "DECNET" must be specified in UPPER CASE. Failing to do so will result in connecting via the DEFAULT method which is to first try connecting via DECnet and if that fails attempt to connect via TCPIP and hence a 75 second stall will take place unless TPC_KEEPINIT is set to a value lower than 150.

9.1.8 Application and Oracle Rdb Both Using SYS$HIBER

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

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

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

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

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

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

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

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

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

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

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

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

9.1.9 Unexpected RCS Termination

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

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

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

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

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

To prevent this problem, Oracle recommends any customer using the Row Cache feature to either avoid defining the logical name RDM$BIND_RCS_VALIDATE_SECS or if this logical name for any reason needs to be defined, to make sure RDM$BIND_RCS_LOG_FILE is correctly defined (i.e. defined with the

/SYSTEM and /EXECUTIVE qualifiers and is pointing to a valid file name in an existing directory on a cluster accessible device with sufficient free space).

This recommendation applies to all versions of Oracle Rdb.

9.1.10 Changes for Processing Existence Logical Names

Oracle Rdb Release 7.2.1.1 changed the handling of so called "existence" logical names used to tune the Rdb environment. These existence logical names could in past versions be defined to any value to enable their effect. The Rdb documentation in most cases described using the value 1 or YES as that value and this change is upward compatible with the documentation.

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

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

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

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

During qualification testing of Oracle Rdb 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 kits (or subsequent replacement, if superseded) prior to using Oracle Rdb on OpenVMS V8.3 systems:

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

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

Bug 2351248

A SQL Module or Pre−compiled SQL program built with Rdb 6.1 or earlier may fail when running under Rdb 7.3 if the program submits queries that involve certain kinds of character operations on parameters in the queries. For example, a LIKE operator in the WHERE clause of a SQL statement requires SQL to look for character− or string−matching wildcard characters. Another example is the use of IGNORE CASE which causes SQL to equivalence upper and lower case characters for the character set in use.

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

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

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

If the SQL Module containing the code above is compiled and linked into an executable using a pre−7.0 version of Rdb, it will run properly against that version. However if the same program is run in an Rdb 7.3 or later environment, a call to the SQL_OPN_NAME_LIST procedure will return a SQLCODE of −1. The RDB$MESSAGE_VECTOR will contain a code associated with the following message:

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

To workaround this problem, re−link the program using a 7.3 or later version of SQL$INT.EXE and/or SQL$USER.OLB.

9.1.13 External Routine Images Linked with PTHREAD$RTL

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

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

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

- TO_CHAR
- TO_NUMBER
- TO_DATE

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

9.1.14 Using Databases from Releases Earlier than V7.0

You cannot convert or restore databases earlier than the Oracle Rdb V7.0 format directly to Oracle Rdb V7.3 format. The RMU Convert command for Oracle Rdb V7.3 supports conversions from Oracle Rdb V7.0, V7.1 and V7.2 format databases only. If you have an Oracle Rdb V3.0 through V6.1 format database, you must convert it to at least Oracle Rdb V7.0 format and then convert it to Oracle Rdb V7.3 format. For example, if you have a V4.2 format database, you must convert it first to at least Oracle Rdb V7.0 format, then convert it to Oracle Rdb V7.3 format.

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

9.1.15 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.

9.1.16 New Attributes Saved by RMU/UNLOAD Incompatible With Prior Versions

Bug 2676851

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

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

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

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

9.1.17 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.

9.1.18 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.

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

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

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

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

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

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

```
SQL> alter table t1 alter column f1
```

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

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

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

or:

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

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

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

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

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

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

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

The strategy for a delete on T2 is now:

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

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

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

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

9.1.20 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.

9.1.21 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.

9.1.22 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.

9.1.23 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 SORT32 run−time library.

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

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

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

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

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

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

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

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

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

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

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

9.1.24 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 9−1 Sort Memory Logicals

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

9.1.25 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.

9.2 SQL Known Problems and Restrictions

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

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

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

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

This restriction does not apply to interactive or dynamic SQL.

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

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

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

9.2.2 Multistatement or Stored Procedures May Cause Hangs

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

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

```
Session 1:

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

Session 2:

$ RMU/BACKUP/LOG/ONLINE MF_PERSONNEL MF_PERSONNEL
```

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

```
$ RMU/SHOW LOCKS /MODE=BLOCKING MF_PERSONNEL
    .
    .
    .
Resource: nowait signal

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

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

9.2.3 Use of Oracle Rdb from Shareable Images

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

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

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

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

9.3 Oracle RMU Known Problems and Restrictions

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

9.3.1 RMU/CONVERT Fails When Maximum Relation ID is Exceeded

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

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

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

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

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

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

9.3.2 RMU/UNLOAD/AFTER_JOURNAL Requires Accurate AIP Logical Area Information

The RMU/UNLOAD/AFTER_JOURNAL command uses the on−disk area inventory pages (AIPs) to determine the appropriate type of each logical area when reconstructing logical dbkeys for records stored in mixed−format storage areas. However, the logical area type information in the AIP is generally unknown for logical areas created prior to Oracle Rdb release 7.0.1. If the RMU/UNLOAD /AFTER_JOURNAL command cannot determine the logical area type for one or more AIP entries, a warning message is displayed for each such area and may ultimately return logical dbkeys with a 0 (zero) area number for records stored in mixed−format storage areas.

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

```
EMPLOYEES /TYPE=TABLE
```

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

```
EMPLOYEES /AREA=EMPID_OVER /TYPE=TABLE
```

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

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

  Note

  This type should NOT be used for the RDB$SYSTEM logical areas. This type does NOT identify system relations.
- BLOB
  Specifies that the logical area is a BLOB repository.

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

9.3.3 Do Not Use HYPERSORT with RMU/OPTIMIZE/AFTER_JOURNAL Command

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

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

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

9.3.4 Changes in EXCLUDE and INCLUDE Qualifiers for RMU/BACKUP

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

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

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

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

9.3.5 RMU/BACKUP Operations Should Use Only One Type of Tape Drive

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

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

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

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

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

9.3.6 RMU/VERIFY Reports PGSPAMENT or PGSPMCLST Errors

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

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

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

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

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

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

- Recreate the database by performing:
    1. SQL EXPORT
    2. SQL DROP DATABASE

    3. SQL IMPORT

- Recreate the database by performing:
    1. RMU/BACKUP
    2. SQL DROP DATABASE
    3. RMU/RESTORE
- Repair the SPAM pages by using the RMU/REPAIR command. Note that the RMU/REPAIR command does not write its changes to an after−image journal (AIJ) file. Therefore, Oracle recommends that a full database backup be performed immediately after using the RMU/REPAIR command.

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

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

9.4.1 Converting Single−File Databases

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

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

9.4.2 Row Caches and Exclusive Access

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

9.4.3 Exclusive Access Transactions May Deadlock with RCS Process

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

There are at least three suggested workarounds to this problem:

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

9.4.4 Strict Partitioning May Scan Extra Partitions

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

```
SQL> create table T1
cont>     (id integer
cont>     ,last_name char(12)
cont>     ,first_name char(12)
cont>     );
SQL> create storage map M for T1
```

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

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

9.4.5 Restriction When Adding Storage Areas with Users Attached to Database

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

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

or

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

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

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

9.4.6 Multiblock Page Writes May Require Restore Operation

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

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

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

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

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

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

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

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

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

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

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

9.5.1 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.)

9.5.2 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.

9.5.3 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 9−2 Elapsed Time for Index Creations

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

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

9.5.4 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.

9.5.5 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.

9.5.6 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