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

Table of Contents

Oracle® Rdb for OpenVMS

Release Notes

Release 7.2.5.1

February 2012

Oracle Rdb Release Notes, Release 7.2.5.1 for OpenVMS

Contents

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb Release 7.2.5.1. The notes describe changed and enhanced features; upgrade and compatibility information; new and existing software problems and restrictions; and software and documentation corrections.

Intended Audience

This manual is intended for use by all Oracle Rdb users. Read this manual before you install, upgrade, or use Oracle Rdb Release 7.2.5.1.

Document Structure

This manual consists of the following chapters:

Chapter 1	Describes how to install Oracle Rdb Release 7.2.5.1.
Chapter 2	Describes problems corrected in Oracle Rdb Release 7.2.5.1.
Chapter 3	Describes problems corrected in Oracle Rdb Release 7.2.5.0.
Chapter 4	Describes enhancements introduced in Oracle Rdb Release 7.2.5.1.
Chapter 5	Describes enhancements introduced in Oracle Rdb Release 7.2.5.0.
Chapter 6	Describes enhancements introduced in Oracle Rdb Release 7.2.4.2.
Chapter 7	Describes enhancements introduced in Oracle Rdb Release 7.2.4.1.
Chapter 8	Describes enhancements introduced in Oracle Rdb Release 7.2.4.0.
Chapter 9	Provides information not currently available in the Oracle Rdb documentation set.
Chapter 10	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb Release 7.2.5.1.

Chapter 1
Installing Oracle Rdb Release 7.2.5.1

This software update is installed using the OpenVMS VMSINSTAL utility.

NOTE

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

1.1 Oracle Rdb on HP OpenVMS Industry Standard 64

The Oracle Rdb product family is available on the HP OpenVMS Industry Standard 64 platform and the OpenVMS AlphaServer platform. In general, the functionality 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 7.2 is the same on both I64 and Alpha platforms and databases may be accessed simultaneously from both architectures in a cluster environment. Access to an Oracle Rdb Release 7.2 database from prior Rdb versions (on Alpha or VAX platforms) or from other systems on the network is available via the Oracle Rdb remote database server.

1.2 Requirements

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

- This Oracle Rdb release requires the following OpenVMS environments:
  - ♦ OpenVMS Alpha V8.2 to V8.4−x.
  - ♦ OpenVMS Industry Standard 64 V8.2−1 to V8.4−x.
- Oracle Rdb must be shutdown before you install this update kit. That is, the command file SYS$STARTUP:RMONSTOP72.COM should be executed before proceeding with this installation. If you have an OpenVMS cluster, you must shutdown the Rdb Release 7.2 monitor on all nodes in the cluster before proceeding.
- After executing RMONSTOP72.COM, no process on any system in the cluster should have any existing RDMSHRP72.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.
- Oracle strongly recommends that all available OpenVMS patches are installed on all systems prior to installing Oracle Rdb. Contact your HP support representitive for more information and assistance.

1.2.1 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 RDMSHRP72.EXE image. This procedure should be executed by a privileged account after RMONSTOP72 has been run. Any processes that have RDMSHRP72.EXE activated at this point should be terminated prior to starting the Rdb installation procedure.

```
TMP
$ 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 RDMSHRP72.EXE,"PID:"
$ SEARCH 'RDB$TMP' RDMSHRP72.EXE /WINDOW=(1,0)
$ DELETE /NOLOG 'RDB$TMP';*{text}
```

In the following example, the process 2729F16D named "FOO$SERVER" has the image RDMSHRP72.EXE activated even after RMONSTOP72.COM has been executed and this process is terminated prior to starting the Rdb installation procedure:

```
$ @SYS$STARTUP:RMONSTOP72.COM
   .
   .
```

```
   .
$ @FIND_RDMSHRP72_PROC.COM

OpenVMS system analyzer

Process index: 016D   Name: FOO$SERVER  Extended PID: 2729F16D
 0240  7FEF4460 8384F300 $1$DGA2:[VMS$COMMON.SYSLIB]RDMSHRP72.EXE;722

$ 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 721. An RMU /CONVERT operation is required for databases created by or accessed by Oracle Rdb V7.0 or V7.1 to be accessed with Rdb Release 7.2.

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

1.6 Using Databases from Releases Earlier than V7.0

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

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

1.7 Invoking the VMSINSTAL Procedure

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

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

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

```
@SYS$UPDATE:VMSINSTAL RDBV72510IM device−name
```

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

```
@SYS$UPDATE:VMSINSTAL RDBV72510AM device−name
```

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

```
@SYS$UPDATE:VMSINSTAL RDBV72511AM device−name
```

device−name

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

1.8 Stopping the Installation

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

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

1.9 After Installing Oracle Rdb

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

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

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

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

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

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

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

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

1.10 VMS$MEM_RESIDENT_USER Rights Identifier Required

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

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

1.11 Installation, Configuration, Migration, Upgrade Suggestions

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

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

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

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; the firth digit inidicating an optimization level is not significant in this requiredment). 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 enviroments containing Integrity Server systems, there are numerous possible paths depending on the requirements of individual sites. In general, this can be as straightforward as adding a new node to an already existing AlphaServer systems cluster or standalone system, except the node is an HP Integrity server. Table 1−1, Migration Suggestions, considers several possible situations and recommended steps to take.

Table 1−1 Migration Suggestions

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

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

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

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

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

Chapter 2
Software Errors Fixed in Oracle Rdb Release 7.2.5.1

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.5.1.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 Unexpected Memory Allocation Failure When Accessing Remote Database

Bug 12760132

In certain cases when accessing a remote database, there could be a small memory leak where memory was allocated and never released. If this situation continued for a significant length of time, Rdb could exhaust the available memory resulting in unusual errors.

Using an explicit SET TRANSACTION or START TRANSACTION contributes to this problem.

This problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.1.2 Alignment Faults on Itanium Using Multiple Mapped Index Columns

Bug 13552628

In prior releases of Oracle Rdb on the Itanium platform, significant numbers of alignment faults in EXEC mode would occur when creating indices with multiple mapped columns. This negatively affected system performance.

```
drop table t1 cascade if exists;
create table t1
 (f1 bigint, f2 integer, f3 integer);
$write sys$output "Start of populating the table."
begin
declare :i integer;
for :i in 1 to 1000000 step 1
do
insert into t1 values (:i, 1, 1);
end for;
end;
$write sys$output "Start of index creation."
create unique index i1 on t1 (f1 asc,
  f2 mapping values 0 to 32767 asc,
  f3 mapping values 0 to 32767 asc)
type is sorted ranked;
```

The number of alignment faults can be observed by using the MONITOR ALIGNMENT command on another screen on the same system.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.1.3 Problem Writing Large TSN Values to Data and Snap Pages

Bug 12741822

A problem has been discovered that could affect how Transaction Sequence Numbers (TSNs) are stored on data and snapshot pages. This problem only occurs when TSNs have a value greater than 2,147,483,647.

A TSN is a sequentially increasing number associated with an Rdb database transaction. A unique TSN is assigned at the start of an update transaction and represents when the transaction started relative to other transactions.

TSNs are used throughout Rdb memory data structures. Additionally, Rdb writes the TSN to on−disk structures such as data files, snapshot files, and after−image journal (AIJ) files to reflect which transaction modified a particular row. Read−only transactions use the TSNs stored on data and snapshot pages to determine which version of a modified row can be seen by the reader.

The problem occurs due to a change to Rdb Version 7.0 in how TSNs are stored on data and snap pages. Prior to that version, TSNs were 32−bit integers with a maximum value of 4,294,967,295. With 7.0, the size of the TSN field was changed to be a 64−bit integer. So as not to require a database unload and reload to support the larger field size, an algorithm was devised to break up those TSNs that would no longer fit into a 32−bit field into 2 pieces when the page is updated. The original value would be regenerated prior to reading from the page.

Recently, it was discovered that this algorithm could cause such large TSNs to possibly be stored incorrectly. Occurrences would be more prevalent during the window where some active transaction would have TSN values greater than the 2,147,483,647 value, while others had values less than that maximum.

The problem may manifest itself with errors during an RMU /VERIFY /ONLINE, or with incorrect data returned from read−only transactions. Additionally, an RMU /BACKUP /ONLINE may produce a backup file that will restore a corrupted database (although an RMU /RECOVER of all spanning after−journal files will usually fix the corruption).

If you believe that your database exhibits these symptoms, you can use the RMU /RESTORE /ONLY_ROOT /SET_TSN command to raise the next available TSN to 6,500,000,000, a sufficiently high value to prevent this problem. As always, Oracle recommends taking a full off−line database backup prior to any rootfile modifications. An example follows:

```
$ RMU/BACKUP mydb.rdb mydb.rbf
$ RMU/RESTORE/ONLY_ROOT/SET_TSN=(TSN=6500000000, CSN=6500000000) mydb.rbf
```

Note

A Support article about this issue and how to resolve it has been written. Please reference document number 1440100.1 on My Oracle Support to read this article. Support provides a command procedure that can patch the TSNBLK and CSNBLK numbers to get around the problem.

The problem has been corrected in Oracle Rdb Release 7.2.5.1. TSNs will now be stored correctly on data pages.

2.1.4 Query With Complex Shared OR Predicates Returns Wrong Result

Bug 12690624

The customer demonstrates, using the following reproducer, that the inner join query with complex OR predicates returns wrong results (3 rows instead of 0 rows).

```
SELECT T2.NSEG, T2.SENS
FROM (((C_AVG T0
        INNER JOIN C_SBT T1 ON T0.C_COD_LIG = T1.C_COD_LIG)
        INNER JOIN C_SGU T2 ON T1.NSEG  = T2.NSEG)
        INNER JOIN C_ETU T3 ON T3.NITIN = T2.NITIN)
        INNER JOIN C_ITI T4 ON T4.NITIN = T3.NITIN
WHERE
  T3.IDAT  = 99999 AND T0.NUMAG = '999 XXX' AND
  -- BLOC 1:
  (T0.DEBU = T0.PFIN AND T0.DEBU >= T1.DEBU AND T0.DEBU <= T1.PFIN) AND
  -- BLOC 3:
  ((SELECT COUNT(*)
      FROM (C_BOUC T5 INNER JOIN C_OBST T6 ON T6.ID_OBST = T5.ID_OBST)
              INNER JOIN C_VOBST T7 ON
                  T7.ID_OBST = T5.ID_OBST AND T7.CLIG = T5.CLIG AND
                  T7.NOMV = T5.NOMV
      WHERE T5.NUMAG = T0.NUMAG AND T5.ACT = 1 AND T6.ACT = 1 AND
        -- BLOC 3a:
        (T6.DEBU = T6.PFIN AND T6.DEBU >= T1.DEBU AND T6.DEBU <= T1.PFIN) AND
        -- BLOC 3b:
         -- Segment A OR
        ((T2.PSEG = 2 AND T2.SENS = 0 AND T6.DEBU >= T4.FITI) OR
         -- Segment B
         (T2.PSEG = 3 AND T6.DEBU <> T6.PFIN AND T2.SENS = 1 AND
          T6.PFIN > T4.DITI AND T6.DEBU < T4.FITI)) AND
        -- BLOC 3c:
        (T3.UTLP = 1 AND
         ((T2.SENS = 0 AND T7.SCIR = 7) OR         ! T2_SENS = 0 is shared
          (T2.SENS = 1 AND T7.SCIR IN (1,3, 8))))  ! T2_SENS = 1 is shared
    ) > 0               -- count > 0
   );
 T2.NSEG   T2.SENS
    2181         0
    2181         0
    2181         0
3 rows selected
```

There is no workaround for this problem.

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

1. The main query is an inner join of five tables, with a WHERE clause containing a COUNT sub−query of 3 tables inner joined.
2. The sub−query contains the complex expression with AND and OR predicates.
3. Some of the operands inside the last OR predicates are shared by the previous OR predicates.

This problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.1.5 Query With Shared OR Predicates Returns Wrong Result

Bug 12632105

A customer demonstrates, using the following simple reproducer, that this query returns wrong results.

```
create database filename testdb
create table tab1 (
       col1 bigint, col2 integer, col3 tinyint, col4 tinyint);
create table tab2 (col1 bigint);
insert into tab1 values (1000000,900000,11,2);
commit;
```

The following query should return no rows but it returns one row.

```
SELECT col2, col3, col4
 from TAB1 T1 where
 col3 in (15,13,11)
        and
   (
      (col3 = 15 and
           ((col4 = 6) or
            (col4 = 2 and
                exists (select * from TAB2 T2
                        where T1.col1 = T2.col1)
             )
            )
       )
   or
     (col2 between 1000000 and 1500000 and
           ((col3 = 11 and col4 = 2) or
            (col3 = 13 and col4 = 1)
           )
      )
     )
 and col1 = 1000000;
Tables:
  0 = TAB1
  1 = TAB2
Cross block of 2 entries  Q1
  Cross block entry 1
    Conjunct: (0.COL3 = 15) OR (0.COL3 = 13) OR (0.COL3 = 11)
    Conjunct: 0.COL1 = 1000000
    Get     Retrieval sequentially of relation 0:TAB1
  Cross block entry 2
    Aggregate-F1: 0:COUNT-ANY (<subselect>) Q2
    Conjunct: 0.COL1 = 1.COL1
    Get     Retrieval sequentially of relation 1:TAB2
        COL2   COL3   COL4
      900000     11      2
1 row selected
```

However, moving "col3 in (15,13,11)" to the end returns the correct result, as seen in the following example.

```
SELECT col2, col3, col4
 from TAB1 T1 where
   (
```

```
      (col3 = 15 and
           ((col4 = 6) or
            (col4 = 2 and
                exists (select * from TAB2 T2
                        where T1.col1 = T2.col1)
             )
            )
       )
   or
     (col2 between 1000000 and 1500000 and
           ((col3 = 11 and col4 = 2) or
            (col3 = 13 and col4 = 1)
           )
      )
     )
 and col1 = 1000000
 and col3 in (15,13,11);
Tables:
  0 = TAB1
  1 = TAB2
Cross block of 2 entries  Q1
  Cross block entry 1
    Conjunct: 0.COL1 = 1000000
    Conjunct: (0.COL3 = 15) OR (0.COL3 = 13) OR (0.COL3 = 11)
    Get     Retrieval sequentially of relation 0:TAB1
  Cross block entry 2

    Conjunct: (((0.COL3 = 15) AND ((0.COL4 = 6) OR ((0.COL4 = 2) AND (<agg0> <>
              0)))) OR ((0.COL2 >= 1000000) AND (0.COL2 <= 1500000) AND (((
              0.COL3 = 11) AND (0.COL4 = 2)) OR ((0.COL3 = 13) AND (0.COL4 = 1))
              ))) AND ((0.COL3 = 15) OR (0.COL3 = 13) OR (0.COL3 = 11))
    Aggregate-F1: 0:COUNT-ANY (<subselect>) Q2
    Conjunct: (0.COL3 = 15) OR (0.COL3 = 13) OR (0.COL3 = 11)
    Conjunct: 0.COL1 = 1.COL1
    Get     Retrieval sequentially of relation 1:TAB2
0 rows selected
```

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

1. The query contains an expression with nested AND and OR predicates.
2. Some of the operands inside the OR predicates are shared (the same expression).
3. The placing of one of the OR predicates as part of the AND operation changes the outcome.

This problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.1.6 Query With LSS, LEQ and NOT NULL Predicate Returns Wrong Result on Itanium System

Bug 12794427

The customer demonstrates, using the following simple reproducer, that a query with LSS, LEQ and NOT NULL returns wrong results.

```
create table T1(f1 char(8), f2 char(4), f3 integer);

insert into T1 values ('03901605' , '1',1);
insert into T1 values ('03901605' , ' ',2);
```

```
insert into T1 values ('03901605' , '    ',3);
insert into T1 values ('03901605' , x'01000000',4);
insert into T1 values ('03901605' ,null,1);

creat index T1_idx1 on T1(f1,f2);

select f1, f2, translate (f2 using rdb$hex), f3 from T1;
 F1         F2                          F3
 03901605   ....   01000000             4
 03901605          20202020             2
 03901605          20202020             3
 03901605   1      31202020             1
 03901605   NULL   NULL                 1
5 rows selected

! should find 4 rows
select f1, f2, translate (f2 using rdb$hex), f3 from T1
where f1 = '03901605' and f2 is not null;
 F1         F2                          F3
 03901605          20202020             2
 03901605          20202020             3
 03901605   1      31202020             1
3 rows selected

! should find 1 row
select f1, f2, translate (f2 using rdb$hex), f3 from T1
where f1 = '03901605' and f2 < '';
0 rows selected

! should find 3 rows
select f1, f2, translate (f2 using rdb$hex), f3 from T1
where f1 = '03901605' and f2 <= '';
 F1         F2                          F3
 03901605          20202020             2
 03901605          20202020             3
2 rows selected

!should find 1 row
select f1, f2, translate (f2 using rdb$hex), f3 from T1
where f1 = '03901605' and f2 < x'20';
0 rows selected

!should find 3 rows
select f1, f2, translate (f2 using rdb$hex), f3 from T1
where f1 = '03901605' and f2 <= x'20';
 F1         F2                          F3
 03901605          20202020             2
 03901605          20202020             3
2 rows selected

!should find 1 row
select f1, f2, translate (f2 using rdb$hex), f3 from T1
where f1 = '03901605' and f2 <= x'01000000';
0 rows selected

!should find 1 row
select f1, f2, translate (f2 using rdb$hex), f3 from T1
where f1 = '03901605' and f2 < x'01';
0 rows selected

!should find 1 row
select f1, f2, translate (f2 using rdb$hex), f3 from T1
```

```
where f1 = '03901605' and f2 <= x'01';
0 rows selected
```

The problem occurs when the values of the data are less than or equal to a space character.

This problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.1.7 SQLSRV−E−PWDEXPIRED Error Restored

Bugs 11831591 and 13039749

In Oracle SQL/Services releases prior to 7.3.0.3, if a user account's password lifetime was set, an attempt to connect after the password expired got the SQLSRV−E−PWDEXPIRED error. When intrusion detection was added in Release 7.3.0.3 of SQL/Services, this error changed to SQLSRV−F−GETACCINF. Therefore, applications were unable to trap expired/lifetime exceeded password errors in order to prompt the user for a new password.

An account's password will expire if the account's /PWDLIFETIME or /EXPIRATION is set and the current date/time is later than the set date/time.

SQLSRV−E−PWDEXPIRED is also returned if either of the PWD_EXPIRED or PWD2_EXPIRED flags are set in the system UAF file.

A new Oracle SQL/Services Kit is NOT required to correct this problem. This problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.1.8 Incorrect Results on IA64 using Partitioned Descending Index

Bug 6129797

Starting with Oracle Rdb Release 7.2.4, it was possible for some queries using partitioned indexes with segments of mixed ascending and descending order to return incorrect results on IA64 systems effected by this problem.

The following examples show two problems when using partitioned indexes 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 );

create index x4 on mesa (id asc , m4 asc) ;

create index y4 on rasa (id asc , r4 desc)
       store using (id, r4)
```

```
       in fooa with limit of (1, 'g' )
       otherwise in foob ;
commit;

! the following query returns correctly 3 rows on Alpha but 1 row on IA64:
SQL> sh version
Current version of SQL is: Oracle Rdb SQL X7.2−00
Underlying versions are:
    Database with filename foo
        Oracle Rdb X7.2−00
        Rdb/Dispatch X7.2−01 (OpenVMS Alpha)

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> sh version
Current version of SQL is: Oracle Rdb SQL X7.2−00
Underlying versions are:
    Database with filename foo
        Oracle Rdb X7.2−00
        Rdb/Dispatch X7.2−01 (OpenVMS IA64)

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
```

This problem is related to the construction and comparison of the descending key values during the index partitions.

The problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.1.9 Unexpected Failure When Identity Sequence is Not Granted Access

Bug 13248799

Oracle Rdb creates a special sequence when the IDENTITY clause is used in a CREATE TABLE or ALTER TABLE statement. However, subsequent use of the GRANT statement on the table would not modify the protections on the matching identity sequence. In such cases, the INSERT statement might fail even when the user was granted INSERT privilege on the table.

The following example shows the reported error. As you can see, it is confusing since the user clearly has INSERT privilege on the table.

```
SQL> show privileges on table T;
Privileges on Table T
    (IDENTIFIER=[RDB,TESTER],ACCESS=SELECT+INSERT)
SQL>
SQL> insert into T default values;
%RDB−E−NO_PRIV, privilege denied by database facility
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1. The GRANT statement will now propagate the SELECT privilege to the identity sequence when INSERT is granted on the table. Likewise, a REVOKE statement will remove the SELECT privilege from the identity sequence when INSERT is revoked from the table.

2.1.10 LIMIT TO/ORDER BY Query With OR Predicate Returns Wrong Result

Bug 13554836

The customer demonstrates, using the following simple reproducer, that the LIMIT TO/ORDER BY query with OR predicate returns the wrong result.

```
select DFID,TID,TT_ID from DF;
     DFID               TID     TT_ID
 15307659         313954196         2
  3571816         803166548         1
 18736151         216931932         1
3 rows selected

select * from NT;
 ITEM_IND           ITEM_ID   USER_ID
 F                   3039532   DARMAWANT
 F                   3571816   YAPT
 F                  18736151   GSHG
 T                  15643968   SYSTEM
 T                  16610114   SYSTEM
5 rows selected

! The value returned in the column NOTES_IND in the first row should be 'Y'
! instead of 'N'.
!
select
    TT.DFID
    ,case
        when
            (exists
                (select
                    *
                from
                    NT N
                where
                    ((N.ITEM_ID = TT.DFID and N.ITEM_IND = 'F')
                or  (N.ITEM_ID = TT.TID and N.ITEM_IND = 'T')
                or  (N.ITEM_ID = TT.PID and N.ITEM_IND = 'M'))
                and N.USER_ID <> 'SYSTEM'))
        then
            'Y'
        else
            'N'
    end NOTES_IND
from
    (select
        DF.DFID,
        DF.TID,
        DF.PID
    from
        DF DF
```

```
    ) TT
order by
   TT.DFID
limit to 150 rows;
       DFID   NOTES_IND
    3571816   N          <= this is wrong
   15307659   N
   18736151   Y
3 rows selected
```

The query works if the LIMIT TO clause is removed, as in the following example.

```
select
    TT.DFID
    ,case
        when
            (exists
                (select
                    *
                from
                    NT N
                where
                    ((N.ITEM_ID = TT.DFID and N.ITEM_IND = 'F')
                or  (N.ITEM_ID = TT.TID and N.ITEM_IND = 'T')
                or  (N.ITEM_ID = TT.PID and N.ITEM_IND = 'M'))
                and N.USER_ID <> 'SYSTEM'))
        then
            'Y'
        else
            'N'
    end NOTES_IND
from
    (select
        DF.DFID,
        DF.TID,
        DF.PID
    from
        DF DF
    ) TT
order by
   TT.DFID
!limit to 150 rows    <= removing this makes the query work
;
Tables:
  0 = DF
  1 = NT
Sort: 0.DFID(a)
Cross block of 2 entries  Q0
  Cross block entry 1
    Merge of 1 entries  Q1
      Merge block entry 1  Q2
      Get     Retrieval sequentially of relation 0:DF
  Cross block entry 2
    Aggregate-F1: 0:COUNT-ANY (<subselect>) Q3
    Conjunct: (((1.ITEM_ID = 0.DFID) AND (1.ITEM_IND = 'F')) OR (
              (1.ITEM_ID = 0.TID) AND (1.ITEM_IND = 'T')) OR ((
              1.ITEM_ID = 0.PID) AND (1.ITEM_IND = 'M'))) AND (
              1.USER_ID <> 'SYSTEM')
    OR index retrieval Q3
      Conjunct: ((1.ITEM_ID = 0.DFID) AND (1.ITEM_IND = 'F')) OR
                ((1.ITEM_ID = 0.TID) AND (1.ITEM_IND = 'T'))
      OR index retrieval Q3
```

```
        Index only retrieval of relation 1:NT
          Index name  NT_IDX [2:2]
            Keys: (1.ITEM_IND = 'F') AND (1.ITEM_ID = 0.DFID)
        Conjunct: NOT ((1.ITEM_IND = 'F') AND (1.ITEM_ID = 0.DFID
                  ))
        Index only retrieval of relation 1:NT
          Index name  NT_IDX [2:2]
            Keys: (1.ITEM_IND = 'T') AND (1.ITEM_ID = 0.TID)
      Conjunct: NOT (((1.ITEM_IND = 'F') AND (1.ITEM_ID = 0.DFID)
                ) OR ((1.ITEM_IND = 'T') AND (1.ITEM_ID = 0.TID)))
      Index only retrieval of relation 1:NT
        Index name  NT_IDX [2:2]
          Keys: (1.ITEM_IND = 'M') AND (1.ITEM_ID = 0.PID)
       DFID   NOTES_IND
    3571816   Y           <= this is correct
   15307659   N
   18736151   Y
3 rows selected
```

The problem occurs when the query contains a nested OR predicate in the inner loop of the cross strategy with ORDER BY followed by LIMIT TO clause.

This problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.2 SQL Errors Fixed

2.2.1 THRESHOLDS Clause Not Applied to Default LIST Storage by CREATE STORAGE MAP Statement

Bug 12664565

In prior releases of Oracle Rdb, the CREATE STORAGE MAP statement did not apply THRESHOLD values to the default LIST storage area when it was referenced in the storage map definition.

The following example shows this problem. The Partition Information for the lists map does not report any THRESHOLDS for the default LIST area.

```
SQL> create storage map LISTS_MAP
cont>     store LISTS
cont>         in AREA_BLOB_1
cont>             (thresholds are (50, 60, 80))
cont>             for (BLOB_1.IMAGE)
cont>         in AREA_BLOB_DEFAULT
cont>             (thresholds are (55, 65, 85));
SQL> commit;
SQL>
SQL> show storage map lists_map
     LISTS_MAP
 For Lists
 Store clause:          STORE LISTS
        in AREA_BLOB_1
            (thresholds are (50, 60, 80))
            for (BLOB_1.IMAGE)
        in AREA_BLOB_DEFAULT
            (thresholds are (55, 65, 85))

 Partition information for lists map:
  Partition: (0) SYS_P00059
   Storage Area: AREA_BLOB_1
         Thresholds are (50, 60, 80)
  Partition: (0) SYS_P00001
   Storage Area: AREA_BLOB_DEFAULT
SQL>
```

The default LIST (or SEGMENTED STRING) area is defined by the CREATE DATABASE statement and is created at that time without thresholds. Later during CREATE STORAGE MAP, this logical area is reused without applying the desired THRESHOLDS.

This problem has been corrected in Oracle Rdb Release 7.2.5.1. This logical area is now altered by the CREATE STORAGE MAP statement to apply the thresholds clause. Note, as with other changes to logical areas, this action is deferred until COMMIT time.

2.2.2 Some CHARACTER SET Clauses Ignored by IMPORT DATABASE Statement

Bug 12740758

In prior versions of Oracle Rdb, the CHARACTER SET clauses specified on the IMPORT DATABASE statement were ignored.

With this release, the following clauses are now processed by IMPORT: DEFAULT CHARACTER SET, NATIONAL CHARACTER SET and IDENTIFIER CHARACTER SET. Their effect is to establish new values for use by new objects created after the IMPORT statement has completed.

The following example shows the problem. The character sets from the original database are retained after the IMPORT.

```
SQL> import database
cont>     from ps_export
cont>     filename ps2
cont>
cont>     default character set ISOLATIN1
cont>     national character set UTF8
cont>     identifier character set DEC_MCS
cont> ;
SQL> show connection RDB$DEFAULT_CONNECTION
Connection: RDB$DEFAULT_CONNECTION
.
.
.
Alias RDB$DBHANDLE:
        Identifier character set is ASCII
        Default character set is WIN_LATIN1
        National character set is UNICODE
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.2.3 Unexpected Bugcheck From DROP INDEX or ALTER INDEX Statements

Bug 12823968

In prior releases of Oracle Rdb, it was possible in rare cases for DROP INDEX, ALTER INDEX ... TRUNCATE PARTITION, ALTER INDEX ... TRUNCATE ALL PARTITIONS, ALTER INDEX ... REBUILD PARTITION, or ALTER INDEX ... REBUILD ALL PARTITIONS to generate a bugcheck similar to that shown below.

- Exception at 0000000081CAA251 : RDMSHRP721\DIO$FETCH_DBKEY + 000003D1
- %SYSTEM−F−ACCVIO, access violation, reason mask=00, virtual address=000000000000034C, PC=0000000081CAA251, PS=00000009
- Saved PC = 0000000080F34860 : RDMSHRP721\PSII2DESTROYDUPCHAINS + 00000430
- Saved PC = 0000000080F33AF0 : RDMSHRP721\PSII2DESTROYTREE + 000005C0
- Saved PC = 0000000080F33A60 : RDMSHRP721\PSII2DESTROYTREE + 00000530
- Saved PC = 000000008130FF40 : RDMSHRP721\RDMS$$CHANGE_INDEX + 00022F20

This problem occurs under the following conditions:

- More than one index exists for the table.
- Two or more of the indices do not include a STORE IN clause, which means these indices are stored in the default storage area by default. It also means that the index nodes from those indices will share a single logical area.
- The index being operated on is one of the indices without a STORE clause.
- The index being operated on is a SORTED RANKED index with many duplicates, enough that the compressed bit map must be split across multiple overflow nodes.
- The NUMBER OF BUFFERS used by the session is too small to contain the full chain of overflow nodes for a single key. This includes any definition of RDM$BIND_BUFFERS.

Under normal circumstances, an index in a UNIFORM logical area is processed by these DROP INDEX and ALTER INDEX statements using the fast logical area delete feature which does not need to traverse the index nodes. Therefore, defining the indices with STORE clauses or using a large NUMBER OF BUFFERS for the database can be used to avoid this problem.

The following example uses SET FLAGS with the INDEX_STATS keyword to display more information during the ALTER INDEX statement.

```
SQL> set flags 'stomap_stats,index_stats';
SQL>
SQL> alter index SAMPLE_INDEX_1 TRUNCATE ALL PARTITIONS;
~Ai alter index "SAMPLE_INDEX_1" (hashed=0, ordered=0)
~As locking table "SAMPLE_TABLE" (PR -> PU)
~Ai truncate all partitions
~Ai truncated 0 partitions, skipped 0
~Ai shared larea 80 - must destroy tree
~Ai destroy old index, root=80:59:1
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]RDSBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=000000000000034C, PC=FFFFFFFF81CDB220, PS=0000001B
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.2.4 Unexpected Error When Both LIKE and COMPRESSION Used in CREATE TABLE Statement

Bug 12907930

In prior releases of Oracle Rdb, attempts to use the COMPRESSION clause and the LIKE clause on a CREATE TABLE or DECLARE LOCAL TEMPORARY TABLE statement caused an exception to be raised.

The following example shows this error.

```
SQL> create table OLD_EMPLOYEES
cont>     like EMPLOYEES
cont>     compression is disabled
cont> ;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-BAD_CODE, corruption in the query string
SQL>
SQL> declare local temporary table module.OLD_EMPLOYEES
```

```
cont>     like EMPLOYEES
cont>     compression is disabled
cont> ;
%RDMS-E-BAD_CODE, corruption in the query string
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1. SQL now generates the correct DDL definition for these combined clauses.

2.2.5 Wrong Results When UNION Mixed With EXCEPT, MINUS or INTERSECT

In prior releases of Oracle Rdb, it was possible that mixing UNION with EXCEPT, MINUS or INTERSECT could return wrong results. The EXCEPT, MINUS or INTERSECT were mistakenly transformed to UNION operators and merged into UNION operations.

The following example should return zero rows but the MINUS is handled incorrectly.

```
SQL> create table X (a int);
SQL> insert into X values (1);
1 row inserted
SQL>
SQL> select a from x
cont> minus
cont> (select a from x
cont> union distinct
cont> select a from x
cont> )
cont> ;
           A
           1
1 row selected
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1. Any SQL Module Language or SQL Precompiler applications that use UNION should be recompiled after this version is installed. Applications using Dynamic or Interactive SQL will automatically get the corrected behavior upon execution of these queries.

2.2.6 Unexpected Error When Defining Trigger With INSERT ... DEFAULT VALUES Clause

In prior releases of Oracle Rdb, it was not possible to use the syntax INSERT INTO ... DEFAULT VALUES as an action in a TRIGGER definition. The syntax was accepted by SQL but the Rdb server rejected the definition as shown below.

```
SQL> create trigger c_insert
cont>     after insert on C
cont>     (trace 'insert into C';
cont>      update A set a = a + 1, b = DEFAULT)
cont>     for each row
cont>
cont>     when (C.a is NULL)
cont>         (insert into B default values)
```

```
cont>     for each row
cont>
cont>     when (C.a = FF (C.a))
cont>         (signal 'RR001' ('Unexpected value'))
cont>     for each row
cont> ;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-BAD_CODE, corruption in the query string
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1. Oracle Rdb now correctly processes this format for the INSERT statement when present in a trigger definition.

2.2.7 Unexpected Bugcheck When Declaring a Local Temporary Table With the Same Name as a System Table

Bug 12907930

In prior versions of Oracle Rdb, it was not permitted to declare a local temporary table with the same name as a system table. In this example, the name RDB$STORAGE_MAPS is a system table which the customer wanted to load into a scratch table and perform comparisons across different databases. The unfortunate side effect is that the table metadata is unloaded as part of the error processing.

The following example shows the resulting error (and bugcheck dump) when the expected system table metadata is missing.

```
SQL> declare local temporary table module.RDB$STORAGE_MAPS
cont>    like RDBVMS$STORAGE_MAPS
cont>    on commit preserve rows;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-VIEW_NO_ANA, views cannot be analyzed
SQL>
SQL> create view VIEW_RDB$STORAGE_MAPS AS SELECT * FROM RDB$STORAGE_MAPS;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]RDSBUGCHK.DMP;
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1. Oracle Rdb now uses the context for the name to correctly cleanup the in−memory metadata.

2.3 RMU Errors Fixed

2.3.1 RMU Extract Did Not Propagate Domain Attributes

Bug 12572092

In prior versions of Oracle Rdb, RMU Extract did not propagate domain attributes (default or check constraint) when the /LANGUAGE=ANSI_SQL or /OPTION=NODOMAIN qualifiers were used.

The following example shows this.

```
SQL> create domain COL1_DOM
cont>     char
cont>     default NULL
cont>     check ((value is null
cont>             or substring(value from 1 for 1) between 'A' and 'Z'))
cont>         not deferrable
cont> ;
SQL>
SQL> create table TBL1
cont>     (col1       COL1_DOM
cont>     )
cont> ;
```

Here is the resulting extracted table when /OPTION=NODOMAIN is used. Note that neither the DEFAULT value for the column nor the CHECK constraint is extracted from the domain.

```
create table TBL1 (
    COL1
        CHAR (1));
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1. RMU Extract now propagates the DEFAULT defined for the domain to each referencing column when extracted using /LANGUAGE=ANSI_SQL or /OPTION=NODOMAIN qualifiers.

RMU Extract now generates a local (column) CHECK constraint with a name derived from the source domain.

```
create table TBL1 (
    COL1
        CHAR (1)
        default NULL
        constraint COL1_DOM_1
            check(((COL1 is null)
                or SUBSTRING(COL1 from 1 for 1) between 'A' and 'Z'))
            not deferrable);
```

Note

The option Defer_Constraints of the /OPTIONS qualifier is not currently applied to the extracted domain constraint.

2.3.2 RMU/RECOVER Consistency Bugcheck When Fetching a SPAM Page

Bug 12639352

During the RMU/RECOVER of an Oracle Rdb database, if, when an AIJBL$K_DELA record in an AIJ file was processd to delete a logical area, no preceeding AIJ record in that AIJ file had previously readied the uniform storage area containing that logical area in Update permission mode and the storage area was not marked as corrupt, a consistency bugcheck would occur when fetching the SPAM page for that logical area. When this happened, the resulting RMUBUGCHK.DMP would show the following stack trace. (The following stack trace is for Oracle Rdb Release 7.2.5.0, but this problem could occur in earlier versions of Oracle Rdb as well.)

```
***** Exception at 0000000080BA0D00 : RMU72\PIO$FETCH + 000005B0
%COSI-F-BUGCHECK, internal consistency failure
Saved PC = 00000000807F5FC0 : RMU72\DIORE$DELA + 000008C0
Saved PC = 00000000807EDBD0 : RMU72\DIO$RE_DO + 00000530
Saved PC = 0000000080B38050 : RMU72\KUTREC$APPLY_AIJBL + 00000B40
Saved PC = 0000000080B2C060 : RMU72\KUTREC$DO_C_AIJBUF + 00001AF0
Saved PC = 0000000080B2A300 : RMU72\KUTREC$DO_COMMIT_QUE + 00000160
Saved PC = 0000000080B3A2B0 : RMU72\KUTREC$NORMAL_ROLLFORWARD + 000003E0
Saved PC = 0000000080B23BB0 : RMU72\KUTREC$RECOVER_JOURNAL + 00003FD0
Saved PC = 0000000080B1E730 : RMU72\KUTREC$RECOVER + 00001E30
Saved PC = 00000000805D0540 : RMU72\RMUREC$RECOVER_ACTION + 00001910
Saved PC = 00000000805CEBA0 : RMU72\RMUCLI$RECOVER + 000009D0
Saved PC = 0000000080332530 : RMU72\RMU_DISPATCH + 00001C30
Saved PC = 0000000080330150 : RMU72\RMU_STARTUP + 00000920
Saved PC = 0000000080000C20 : RMU72\ELF$TFRADR + 00000050
Saved PC = FFFFFFFF844CEE80 : Image PTHREAD$RTL + 0004AE80
Saved PC = FFFFFFFF844846C0 : Image PTHREAD$RTL + 000006C0
Saved PC = 000000007ADBF040 : Image DCL + 0006D040
```

This problem has been fixed. Now the storage area, which contains the SPAM page for the logical area being deleted, will be readied in Update mode if it has not previously been readied during the RMU/RECOVER. There is no workaround for this problem.

The following example shows the problem. First the database is restored. Then during the RMU/RECOVER an internal consistency failure occurs, the recover operation fails, and an RMUBUGCHK.DMP is created.

```
$ RMU/RESTORE/DIR=DEVICE:[DIRECTORY]/NOCDD [-]DB_NAME.RBF
%RMU-I-AIJRSTAVL, 0 after-image journals available for use
%RMU-I-AIJISOFF, after-image journaling has been disabled
%RMU-W-USERECCOM, Use the RMU Recover command. The journals are not available.
$ RMU/RECOVER/ROOT=DEVICE:[DIRECTORY]DB_NAME.RDB/TRACE [-]DB_NAME_01.AIJ
%RMU-I-LOGRECDB, recovering database file DEVICE:[DIRECTORY]DB_NAME.RDB;1
%RMU-I-LOGRECSTAT, transaction with TSN 0:16005673 committed
%RMU-I-LOGRECSTAT, transaction with TSN 0:16005674 committed
%RMU-I-LOGRECSTAT, transaction with TSN 0:16005675 committed
%RMU-I-LOGRECSTAT, transaction with TSN 0:16005676 committed
%RMU-I-LOGRECSTAT, transaction with TSN 0:16005677 committed
%RMU-I-LOGRECSTAT, transaction with TSN 0:16005678 committed
%RMU-I-LOGRECSTAT, transaction with TSN 0:16005679 committed
%COSI-F-BUGCHECK, internal consistency failure
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-I-BUGCHKDMP, generating bugcheck dump file DEVICE:[DIRECTORY]RMUBUGCHK.DMP;
```

The following example shows that this problem has been fixed. Following the RMU/RESTORE of the database, the RMU/RECOVER operation succeeds and a success message is output.

```
$ RMU/RESTORE/DIR=DEVICE:[DIRECTORY]/NOCDD [−]DB_NAME.RBF
%RMU−I−AIJRSTAVL, 0 after−image journals available for use
%RMU−I−AIJISOFF, after−image journaling has been disabled
%RMU−W−USERECCOM, Use the RMU Recover command. The journals are not available.
$ RMU/RECOVER/ROOT=DEVICE:[DIRECTORY]DB_NAME.RDB/TRACE [−]DB_NAME_01.AIJ
%RMU−I−LOGRECDB, recovering database file DEVICE:[DIRECTORY]DB_NAME.RDB;1
%RMU−I−LOGRECSTAT, transaction with TSN 0:16005673 committed
%RMU−I−LOGRECSTAT, transaction with TSN 0:16005674 committed
%RMU−I−LOGRECSTAT, transaction with TSN 0:16005675 committed
%RMU−I−LOGRECSTAT, transaction with TSN 0:16005676 committed
%RMU−I−LOGRECSTAT, transaction with TSN 0:16005677 committed
%RMU−I−LOGRECSTAT, transaction with TSN 0:16005678 committed
%RMU−I−LOGRECSTAT, transaction with TSN 0:16005679 committed
%RMU−I−AIJONEDONE, AIJ file sequence 133 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 134
%RMU−I−AIJNOENABLED, after−image journaling has not yet been enabled
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.3.3 Problems If a Full RMU/BACKUP Was Not Done After RMU/MOVE_AREA

Bug 13070493

Problems occurred if a full database backup of an Oracle Rdb database was not done following an RMU/MOVE_AREA command. If a full database backup that was done previous to the RMU/MOVE_AREA command was later restored, followed by the restore of an incremental database backup done after the RMU/MOVE_AREA command, changes to the database root file made by the RMU/MOVE_AREA command were lost. This would undo the results of the RMU/MOVE_AREA command and sometimes cause database corruption. To prevent this, the following error will now be returned if the next online or offline database backup command executed following an RMU/MOVE_AREA command will not produce a full database backup, and the backup will not be allowed.

```
$ rmu/backup/nolog/incremental mf_personnel.rdb mfp.rbf
%RMU−F−NOFULLBCK, no full backup of this database exists
%RMU−F−FTL_BCK, Fatal error for BACKUP operation at 11−OCT−2011
12:02:18.08
```

If an RMU/DUMP/HEADER command is executed following an RMU/MOVE_AREA command, and a full database backup has not yet been executed, the following warning line will appear in the "Database Backup" section of the dump.

```
    Database Backup...

      − Incremental backup not allowed until full backup
```

Once a full online or offline database backup is executed following the RMU/MOVE_AREA command, incremental database backups can be executed and an RMU/DUMP HEADER command will no longer show

the above warning line in the "Database Backup" section of the dump. Note that Rdb already requires a full database backup to be made following certain changes made to the database root such as adding and reserving storage areas and defining new nodes and users. These existing cases also cause the incremental backup to fail with the same error message and the above warning line to appear in the output of the RMU/DUMP/HEADER command until a full database backup is executed.

The following example shows the problem. The full database backup is done before the move of the DEPARTMENTS storage area to the new directory. The incremental database backup is done after the move. When the full restore is done, the database root is restored as it was before the move was done. This does not corrupt the database in this particular case but it does undo the move of the DEPARTMENTS storage area to the new directory.

```
$ rmu/backup/online/nolog mf_personnel mf_personnel_full
$ SQL
attach 'filename mf_personnel';
insert into departments values ('1234','Testing A','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1235','Testing B','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1236','Testing C','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1237','Testing D','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1238','Testing E','11111',20000,20000);
1 row inserted
commit;
exit;
$ RMU/MOVE_AREA/DIRECTORY=[.move_db]/NOLOG -
  device:[directory]mf_personnel.rdb -
  DEPARTMENTS
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
%RMU-I-COMPLETED, MOVE_AREA operation completed at 11-OCT-2011 12:07:15.40
$ dir departments.*
%DIRECT-W-NOFILES, no files found
$ dir [.move_db]departments.*

Directory DEVICE:[DIRECTORY.MOVE_DB]

DEPARTMENTS.RDA;1   DEPARTMENTS.SNP;1

Total of 2 files.
$ rmu/verify/nolog mf_personnel
$ SQL
attach 'filename mf_personnel';
insert into departments values ('1231','Testing F','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1230','Testing G','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1232','Testing H','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1233','Testing I','11111',20000,20000);
1 row inserted
commit;
```

```
insert into departments values ('1240','Testing J','11111',20000,20000);
1 row inserted
commit;
exit;
$ rmu/backup/online/nolog/incremental mf_personnel mf_personnel_inc
$ SQL
drop database filename 'mf_personnel';
exit;
$ rmu/restore/norecover/nocdd/nolog mf_personnel_full
%RMU-I-AIJRSTAVL, 0 after-image journals available for use
%RMU-I-AIJISOFF, after-image journaling has been disabled
$ rmu/restore/incremental/nocdd/norecover/nolog mf_personnel_inc
DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1,
 restore incrementally? [N]:Y
$ dir departments.*

Directory DEVICE:[DIRECTORY]

DEPARTMENTS.RDA;1   DEPARTMENTS.SNP;1

Total of 2 files.
$ dir [.move_db]departments.*
%DIRECT-W-NOFILES, no files found
$ rmu/verify/nolog mf_personnel
```

The following example shows the new behavior. The full database backup is done before the move of the DEPARTMENTS storage area to the new directory. When the incremental database backup is attempted after the move, the incremental backup is aborted and an error that states that a full backup should have been done after the RMU/MOVE is output.

```
$ rmu/backup/online/nolog mf_personnel mf_personnel_full
$ SQL
attach 'filename mf_personnel';
insert into departments values ('1234','Testing A','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1235','Testing B','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1236','Testing C','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1237','Testing D','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1238','Testing E','11111',20000,20000);
1 row inserted
commit;
exit;
$ RMU/MOVE_AREA/DIRECTORY=[.move_db]/NOLOG -
  device:[directory]mf_personnel.rdb -
  DEPARTMENTS
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
%RMU-I-COMPLETED, MOVE_AREA operation completed at 11-OCT-2011 12:09:15.40
$ dir departments.*
%DIRECT-W-NOFILES, no files found
$ dir [.move_db]departments.*

Directory DEVICE:[DIRECTORY.MOVE_DB]

DEPARTMENTS.RDA;1   DEPARTMENTS.SNP;1
```

```
Total of 2 files.
$ rmu/verify/nolog mf_personnel
$ SQL
attach 'filename mf_personnel';
insert into departments values ('1231','Testing F','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1230','Testing G','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1232','Testing H','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1233','Testing I','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1240','Testing J','11111',20000,20000);
1 row inserted
commit;
exit;
$ rmu/backup/online/nolog/incremental mf_personnel mf_personnel_inc
%RMU-F-NOFULLBCK, no full backup of this database exists
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 11-OCT-2011
12:30:18.08
```

The following example shows that no problems occur if the full database backup is done after the database move instead of before the database move.

```
$ SQL
attach 'filename mf_personnel';
insert into departments values ('1234','Testing A','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1235','Testing B','11111',20000,20000);
1 row inserted
`insert into departments values ('1236','Testing C','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1237','Testing D','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1238','Testing E','11111',20000,20000);
1 row inserted
commit;
exit;
$ RMU/MOVE_AREA/DIRECTORY=[.move_db]/NOLOG -
  device:[directory]mf_personnel.rdb -
  DEPARTMENTS
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
%RMU-I-COMPLETED, MOVE_AREA operation completed at 11-OCT-2011 12:40:15.40
$ dir departments.*
%DIRECT-W-NOFILES, no files found
$ dir [.move_db]departments.*

Directory DEVICE:[DIRECTORY.MOVE_DB]

DEPARTMENTS.RDA;1   DEPARTMENTS.SNP;1

Total of 2 files.
$ rmu/verify/nolog mf_personnel
```

```
$ rmu/backup/online/nolog mf_personnel mf_personnel_full
$ SQL
attach 'filename mf_personnel';
insert into departments values ('1231','Testing F','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1230','Testing G','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1232','Testing H','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1233','Testing I','11111',20000,20000);
1 row inserted
commit;
insert into departments values ('1240','Testing J','11111',20000,20000);
1 row inserted
commit;
exit;
$ rmu/backup/online/nolog/incremental mf_personnel mf_personnel_inc
$ SQL
drop database filename 'mf_personnel';
exit;
$ rmu/restore/norecover/nocdd/nolog mf_personnel_full
%RMU-I-AIJRSTAVL, 0 after-image journals available for use
%RMU-I-AIJISOFF, after-image journaling has been disabled
$ rmu/restore/incremental/nocdd/norecover/nolog mf_personnel_inc
DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1,
 restore incrementally? [N]:Y
$ dir departments.*
%DIRECT-W-NOFILES, no files found
$ dir [.move_db]departments.*

Directory DEVICE:[DIRECTORY.MOVE_DB]

DEPARTMENTS.RDA;1   DEPARTMENTS.SNP;1

Total of 2 files.
$ rmu/verify/nolog mf_personnel
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.3.4 Parallel Incremental Backup RMU−F−NOFULLBCK Error Handling Problem

Bug 13241902

Starting with Oracle Rdb Release 7.2.5.0, the "RMU−F−NOFULLBCK" error is output if certain changes have been made to a database, such as adding a new storage area, that require a full database backup and the user attempts to execute an incremental database backup before doing a full database backup. Due to an error handling problem, for a parallel incremental backup the error message "SYSTEM−E−QUOTA" was put out in place of the "RMU−F−NOFULLBCK" error. This has been corrected and the correct "RMU−F−NOFULLBCK" error will now be output. This was only a problem with parallel incremental database backups. Non parallel database backups correctly put out the "RMU−F−NOFULLBCK" error. This was not a process quota problem but an error handling problem.

The following example shows the problem. An MF_PERSONNEL database is created and a new storage area is added to the database. When a parallel incremental database backup is done and no full database backup has been done since the new storage area was added to the database, an incorrect "SYSTEM−E−QUOTA" message is output instead of a "RMU−F−NOFULLBCK" error.

```
$ @sql$sample:personnel sql m nocdd "" []
$ SQL
alter database file mf_personnel
reserve 3 storage areas;
alter data file mf_personnel
add storage area n3;
exit
$ RMU/BACKUP/PARALLEL=(EXEC=2 ) −
/LIST_PLAN=(TEST.PLAN)/NOEXECUTE/INCLUDE=(RDB$SYSTEM, −
EMPIDS_LOW, EMPIDS_MID, EMPIDS_OVER, SALARY_HISTORY, EMP_INFO, −
DEPARTMENTS,  JOBS,     RESUMES,     RESUME_LISTS, −
N3 ) −
/noexec         −
/incremental            −
/CHECKSUM_VERIFICATION −
/DISK_file      −
MF_PERSONNEL  dev:[current_dir.BACKUP_DIR]back,dev:[current_dir.BACKUP_DIR]
$ rmu /backup /plan  test.plan
%SYSTEM−E−EXQUOTA, process quota exceeded
%RMU−F−FTL_BCK, Fatal error for BACKUP operation at  2−NOV−2011 22:03:39.15
```

The following example shows that this problem is now fixed. An MF_PERSONNEL database is created and a new storage area is added to the database. When a parallel incremental database backup is done and no full database backup has been done since the new storage area was added to the database, the correct "RMU−F−NOFULLBCK" error is output.

```
$ @sql$sample:personnel sql m nocdd "" []
$ SQL
alter database file mf_personnel
reserve 3 storage areas;
alter data file mf_personnel
add storage area n3;
exit
$ RMU/BACKUP/PARALLEL=(EXEC=2 ) −
/LIST_PLAN=(TEST.PLAN)/NOEXECUTE/INCLUDE=(RDB$SYSTEM, −
EMPIDS_LOW, EMPIDS_MID, EMPIDS_OVER, SALARY_HISTORY, EMP_INFO, −
DEPARTMENTS,  JOBS,     RESUMES,     RESUME_LISTS, −
N3 ) −
/noexec         −
/incremental            −
/CHECKSUM_VERIFICATION −
/DISK_file      −
MF_PERSONNEL  dev:[current_dir.BACKUP_DIR]back,dev:[current_dir.BACKUP_DIR]
$ rmu /backup /plan  test.plan
%RMU−F−NOFULLBCK, no full backup of this database exists
%RMU−F−FTL_BCK, Fatal error for BACKUP operation at  2−NOV−2011 22:10:20.10
```

To avoid getting an error, execute a full database backup after making structural changes to the database and then execute an incremental database backup. To get the correct RMU−F−NOFULLBCK error, execute a non parallel incremental database backup.

This problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.3.5 Problem with RMU/REPAIR/INIT=TSNS When TSNs Exceed 4,294,967,295

A problem has been discovered when using the RMU/REPAIR/INIT=TSNS command to initialize TRANSACTION SEQUENCE NUMBERS (TSNs) when the values are greater than 4,294,967,295. No error is returned by RMU, however any such TSN value stored on data or snapshot pages will not be initialized correctly. The result is that the TSN values on the database page will contain a value which is inconsistent with TSNs stored in the database root structures. TSNs contained in these database root structures are initialized correctly.

TSNs are used by the database engine for a variety of purposes. If the values on the database pages are incorrect or inconsistent, this could cause read−only transactions (or online backups) to determine that the wrong version of a record is visible. Also, an incorrect TSN could inhibit the reuse of snapshot pages, causing snapshot files to grow.

This problem was introduced in Oracle Rdb V7.0−000 and has now been fixed in Oracle Rdb Release 7.2.5.1. RMU/REPAIR/INIT=TSNS will now correctly initialize TSNs of any value. If you believe that you've been subject to this problem when using RMU/REPAIR, you should re−execute the RMU command after upgrading to this version.

2.3.6 Incorrect RMU/BACKUP/AFTER Truncate AIJ File Error Handling

Bug 13106530

When the Oracle Rdb RMU/BACKUP/AFTER command is finished backing up a single extensible After Image Journal (AIJ) file, it truncates the AIJ file to the beginning so that the AIJ file no longer contains the database transaction data written to the backup file and new data written to the AIJ file will start at the begining of the truncated AIJ file. During the database backup, transaction data can continue to be added to the extensible AIJ file before and after the truncation operation.

If a non−Rdb process accesses the extensible AIJ file during the RMU/BACKUP/AFTER truncation operation, a file access conflict can occur which will cause the AIJ backup to be aborted. There was an RMU/BACKUP/AFTER error handling problem that caused a loss of some of the transaction data being written to the AIJ file during and after the AIJ backup if a file access conflict occurred during the AIJ file truncation operation which caused the backup to be aborted. When the RMU/BACKUP/AFTER command was repeated, RMU/BACKUP/AFTER would detect at the start that the AIJ truncation operation had failed during the previous AIJ backup and immediately repeat the truncation operation. However, even though this time the truncation operation succeeded, the truncation operation was not done at the correct point and transaction data that should have been backed up was lost. This caused loss of transaction data when an RMU/RECOVER of the database was later done using the backed up AIJ files.

This problem has been fixed and now no loss of data will occur if the RMU/BACKUP/AFTER of an extensible AIJ file fails during the AIJ file truncation operation and the RMU/BACKUP/AFTER of the AIJ file is repeated once the file access conflict has been resolved or has ended. If the file truncation operation fails, it will be repeated at fixed intervals for a fixed period of time. If any of these repeated truncation operations succeeds, the AIJ backup will continue without error. If the file access conflict is of a longer duration, the RMU/BACKUP/AFTER will be aborted with a fatal file access conflict error and the invalid AIJ backup file will be deleted. Then, if the file access conflict is resolved or ends and the

RMU/BACKUP/AFTER command is repeated, a valid AIJ backup file will be created and when an RMU/RECOVER of the database is done no transaction data will be lost.

The following example shows the problem. During the first backup of the single extensible AIJ file, a file access conflict occurs on the AIJ file when the AIJ file is being truncated and the backup operation is aborted. When the file access conflict has been resolved or has ended and the AIJ backup is repeated, RMU/BACKUP/AFTER detects the truncation failure on the previous AIJ backup and repeats the AIJ file truncation. Even though this time the AIJ file truncation succeeds, the %RMU−I−EMPTYAIJ informational message shows that the truncation has not been done correctly and transaction data contained in the AIJ file has been lost.

```
$ RMU/BACKUP/AFTER/NOQUIET/LOG TESTDB ""
%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 AIJ_1 at 05:28:23.11
%RMU−I−LOGCREBCK, created backup file
            DEVICE:[DIRECTORY]AIJ_1_BCK0_18102011.AIJ;1
%RMU−I−LOGAIJBCK, backed up 3 committed transactions at 05:28:50.92
%RMU−I−AIJMODSEQ, next AIJ file sequence number will be 1
%RMU−I−AIJBCKSTOP, backup of after−image journal AIJ_1 did not complete
%RMU−I−OPERNOTIFY, system operator notification: AIJ manual backup
      operation failed
%RMU−F−FILACCERR, error truncating file
−SYSTEM−W−ACCONFLICT, file access conflict
%RMU−F−FTL_BCK, Fatal error for BACKUP operation at 18−OCT−2011 05:28:50.93
$
$ RMU/BACKUP/AFTER/NOQUIET/LOG TESTDB ""
%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 AIJ_1 at 05:32:33.21
 the %RMU−W−BADAIJBCK, previous AIJ backup did not complete
%RMU−I−EMPTYAIJ, after−image journal file is empty
%RMU−I−OPERNOTIFY, system operator notification: AIJ backup
          operation completed
%RMU−I−AIJBCKEND, after−image journal backup operation completed successfully
```

The following example shows that this problem has been fixed. On the first AIJ backup, the file access conflict occurs during the backup truncation operation but it is of short duration so when the new retry logic repeats the truncation operation it succeeds and the AIJ backup completes normally without any loss of AIJ transaction data. On the second AIJ backup, the file access conflict again occurs when the AIJ file is being truncated but it is of longer duration so the repeated retries of the truncation operation all fail, the backup has to be aborted and the invalid AIJ backup file is deleted. When the third AIJ backup is done, the file access conflict does not occur. The third AIJ backup detects that the previous AIJ backup failed but now the backup file truncation operation is done correctly, the backup operation is successful and a valid AIJ backup file is created.

```
$ RMU/BACKUP/AFTER/NOQUIET/LOG TESTDB ""
%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 AIJ_1 at 09:20:11.12
%RMU−I−LOGCREBCK, created backup file
DEVICE:[DIRECTORY]AIJ_1_BCK0_08122011.AIJ;1
%RMU−I−LOGAIJBCK, backed up 3 committed transactions at 09:20:18.63
```

```
%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 09:20:29.03
%RMU-I-LOGAIJBLK, backed up 415615 after-image journal blocks at
09:20:29.03
%RMU-I-LOGAIJBCK, backed up 3 committed transactions at 09:20:29.03
$
$ RMU/BACKUP/AFTER/NOQUIET/LOG TESTDB ""
%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 AIJ_1 at 09:59:02.02
%RMU-I-LOGCREBCK, created backup file
DEVICE:[DIRECTORY]AIJ_1_BCK0_08122011.AIJ;1
%RMU-I-LOGAIJBCK, backed up 3 committed transactions at 09:59:09.96
%RMU-W-DELBCKFIL, backup aborted, deleting backup file
DEVICE:[DIRECTORY]AIJ_1_BCK0_08122011.AIJ;1
%RMU-I-LOGDELFIL, deleted file
DEVICE:[DIRECTORY]AIJ_1_BCK0_08122011.AIJ;1
%RMU-I-AIJBCKSTOP, backup of after-image journal AIJ_1 did not
complete
%RMU-I-OPERNOTIFY, system operator notification: AIJ manual backup
operation failed
%RMU-F-FILACCERR, error truncating after-image journal file
-SYSTEM-W-ACCONFLICT, file access conflict
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at  8-DEC-2011
09:59:10.13
$
$ RMU/BACKUP/AFTER/NOQUIET/LOG TESTDB ""
%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 AIJ_1 at 10:16:41.55
%RMU-W-BADAIJBCK, previous AIJ backup did not complete
%RMU-I-LOGCREBCK, created backup file
DEVICE:[DIRECTORY]AIJ_1_BCK0_08122011.AIJ;1
%RMU-I-LOGAIJBCK, backed up 4 committed transactions at 10:16:50.44
%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 10:16:51.01
%RMU-I-LOGAIJBLK, backed up 559815 after-image journal blocks at
10:16:51.01
%RMU-I-LOGAIJBCK, backed up 4 committed transactions at 10:16:51.01
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.3.7 Unexpected RMU-W-DATNOTIDX Reported by RMU Verify for Rdb$WORKLOAD Table

In prior releases of Oracle Rdb, it was possible to have rows inserted into Rdb$WORKLOAD that had no matching index entries. This may occur if WORKLOAD COLLECTION IS ENABLED and multiple

duplicate workload rows are inserted by different concurrent processes.

In general, this is harmless to the functioning of the database and the usage of WORKLOAD data. However, RMU/VERIFY may report errors similar to the following:

```
%RMU-W-DATNOTIDX, Row in table RDB$WORKLOAD is not in any indexes.
                  Logical dbkey is 60:296:24.
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1. The INSERT of the Rdb$WORKLOAD rows has been corrected so that this error no longer occurs. However, if these duplicate rows already exist then they can be removed using the following process.

1. Save the existing Rdb$WORKLOAD rows.

   ```
   $ RMU/EXTRACT/ITEM=WORKLOAD/OUT=WRKLD.COM MYDATABASE
   ```

2. Truncate the Rdb$WORKLOAD table.

   ```
   SQL> attach 'filename MYDATABASE';
   SQL> truncate table Rdb$WORKLOAD;
   SQL> commit;
   ```

   Note

   This requires Oracle Rdb Release 7.2.5.1 or later which supports using TRUNCATE TABLE for this system table Rdb$WORKLOAD.

   ---

3. Finally execute the extract .COM file (WRKLD.COM in this example) to re-add the older definitions.

   ```
   $ @WRKLD.COM
   ```

2.4 LogMiner Errors Fixed

2.4.1 RMU/UNLOAD/AFTER_JOURNAL SYSTEM−W−ENDOFFILE Error on a Work File

Bug 12646877

When extracting large database transactions, the Oracle Rdb RMU/UNLOAD/AFTER_JOURNAL command creates temporary work files. There was a problem managing the buffer pools used with these work files which could cause a SYSTEM−W−ENDOFFILE error to occur when reading a work file when multiple database tables were being unloaded in the same RMU/UNLOAD/AFTER_JOURNAL command. This error would cause the command to abort the unload operation.

```
%RMU−F−FILACCERR, error reading work file
DEVICE:[DIRECTORY]U2J4IN470PA101HE0280.TMP;1
−SYSTEM−W−ENDOFFILE, end of file
%RMU−F−FTL_RMU, Fatal error for RMU operation at 7−JUL−2011 14:18:19.67
```

This problem has been fixed. The workaround for this problem in earlier versions of Oracle Rdb is to not unload all the database tables in the same RMU/UNLOAD/AFTER_JOURNAL command.

The following example shows the problem and the workaround. The problem occurs when three tables are unloaded in the same RMU/UNLOAD/AFTER_JOURNAL command. The problem does not happen if two tables are unloaded in one RMU/UNLOAD/AFTER_JOURNAL command and one table is unloaded in a second RMU/UNLOAD/AFTER_JOURNAL command. Three RMU/UNLOAD/AFTER_JOURNAL commands, each unloading one table, could also have been used as a workaround.

```
$ RMU/UNLOAD/AFTER_JOURNAL/INCL=ACT=(NOCOMMIT)/STAT=1800 −
/RESTORE_METADATA=TESTRDB_MULTI.METADATA −
/table=(name=T1,output=rdb_logminer_output_file) −
/table=(name=T2,output=rdb_logminer_output_file) −
/table=(name=T3,output=rdb_logminer_output_file) −
/restart=1−28−1−161750−25853−25853 /log −
TESTRDB_MULTI.RDB TESTRDB_PDE.AIJ
%RMU−I−LMMFRDCNT, Read 427 objects from metadata file
"DEVICE:[DIRECTORY]TESTRDB_MULTI.METADATA;1"
%RMU−I−UNLAIJFL, Unloading table T1 to
DEVICE:[DIRECTORY]RDB_LOGMINER_OUTPUT_FILE.DAT;57
%RMU−I−UNLAIJFL, Unloading table T2 to
DEVICE:[DIRECTORY]RDB_LOGMINER_OUTPUT_FILE.DAT;57
%RMU−I−UNLAIJFL, Unloading table T3 to
DEVICE:[DIRECTORY]RDB_LOGMINER_OUTPUT_FILE.DAT;57
%RMU−I−LOGOPNAIJ, opened journal file
DEVICE:[DIRECTORY]TESTRDB_PDE.AIJ;1 at  7−JUL−2011 14:18:18.58
%RMU−I−AIJRSTSEQ, journal sequence number is "1"
%RMU−F−FILACCERR, error reading work file
DEVICE:[DIRECTORY]U2J4IN470PA101HE0280.TMP;1
−SYSTEM−W−ENDOFFILE, end of file
%RMU−F−FTL_RMU, Fatal error for RMU operation at  7−JUL−2011 14:18:19.67
$ RMU/UNLOAD/AFTER_JOURNAL/INCL=ACT=(COMMIT)/STAT=1800 −
/RESTORE_METADATA=TESTRDB_MULTI.METADATA −
/table=(name=T1,output=rdb_logminer_output_file) −
/table=(name=T3,output=rdb_logminer_output_file) −
/restart=1−28−1−161750−25853−25853 /log −
```

```
TESTRDB_MULTI.RDB TESTRDB_PDE.AIJ
%RMU-I-LMMFRDCNT, Read 427 objects from metadata file
"DEVICE:[DIRECTORY]TESTRDB_MULTI.METADATA;1"
%RMU-I-UNLAIJFL, Unloading table T1 to
DEVICE:[DIRECTORY]RDB_LOGMINER_OUTPUT_FILE.DAT;58
%RMU-I-UNLAIJFL, Unloading table T3 to
DEVICE:[DIRECTORY]RDB_LOGMINER_OUTPUT_FILE.DAT;58
%RMU-I-LOGOPNAIJ, opened journal file
DEVICE:[DIRECTORY]TESTRDB_PDE.AIJ;1 at  7-JUL-2011 14:19:23.36
%RMU-I-AIJRSTSEQ, journal sequence number is "1"
%RMU-I-AIJMODSEQ, next AIJ file sequence number will be 2
%RMU-I-LOGSUMMARY, total 6299 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
$ RMU/UNLOAD/AFTER_JOURNAL/INCL=ACT=(COMMIT)/STAT=1800 -
/RESTORE_METADATA=TESTRDB_MULTI.METADATA -
/table=(name=T2,output=rdb_logminer_output_file) -
/restart=1-28-1-161750-25853-25853 /log -
TESTRDB_MULTI.RDB TESTRDB_PDE.AIJ
%RMU-I-LMMFRDCNT, Read 427 objects from metadata file
"DEVICE:[DIRECTORY]TESTRDB_MULTI.METADATA;1"
%RMU-I-UNLAIJFL, Unloading table T2 to
DEVICE:[DIRECTORY]RDB_LOGMINER_OUTPUT_FILE.DAT;59
%RMU-I-LOGOPNAIJ, opened journal file
DEVICE:[DIRECTORY]TESTRDB_PDE.AIJ;1 at  7-JUL-2011 14:20:31.30
%RMU-I-AIJRSTSEQ, journal sequence number is "1"
%RMU-I-AIJMODSEQ, next AIJ file sequence number will be 2
%RMU-I-LOGSUMMARY, total 6299 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
```

The following example shows that this problem has been fixed. Now all the tables can be unloaded in the same RMU/UNLOAD/AFTER_JOURNAL command and the command completes with no error.

```
$ RMU/UNLOAD/AFTER_JOURNAL/INCL=ACT=(NOCOMMIT)/STAT=1800 -
/RESTORE_METADATA=TESTRDB_MULTI.METADATA -
/table=(name=T1,output=rdb_logminer_output_file) -
/table=(name=T2,output=rdb_logminer_output_file) -
/table=(name=T3,output=rdb_logminer_output_file) -
/restart=1-28-1-161750-25853-25853 /log -
TESTRDB_MULTI.RDB TESTRDB_PDE.AIJ
%RMU-I-LMMFRDCNT, Read 427 objects from metadata file
"DEVICE:[DIRECTORY]TESTRDB_MULTI.METADATA;1"
%RMU-I-UNLAIJFL, Unloading table T1 to
DEVICE:[DIRECTORY]RDB_LOGMINER_OUTPUT_FILE.DAT;56
%RMU-I-UNLAIJFL, Unloading table T2 to
DEVICE:[DIRECTORY]RDB_LOGMINER_OUTPUT_FILE.DAT;56
%RMU-I-UNLAIJFL, Unloading table T3 to
DEVICE:[DIRECTORY]RDB_LOGMINER_OUTPUT_FILE.DAT;56
%RMU-I-LOGOPNAIJ, opened journal file
DEVICE:[DIRECTORY]TESTRDB_PDE.AIJ;1 at  7-JUL-2011 14:15:47.64
%RMU-I-AIJRSTSEQ, journal sequence number is "1"
%RMU-I-AIJMODSEQ, next AIJ file sequence number will be 2
%RMU-I-LOGSUMMARY, total 6299 transactions committed
%RMU-I-LOGSUMMARY, total 0 transactions rolled back
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.5 RMU Show Statistics Errors Fixed

2.5.1 RMU/SHOW STATISTICS Configuration File Problems in Oracle Rdb Release 7.2.5.0

Bugs 12710800 and 12710931

The Oracle Rdb RMU/SHOW STATISTICS implementation for Configuration File support did not work correctly in Oracle Rdb Release 7.2.5.0. This problem did not affect other RMU/SHOW STATISTICS functions, just the functions related to the Configuration File. The workaround for this problem is not to use the Configuration File with RMU/SHOW STATISTICS for Release 7.2.5.0 but to use the equivalent command line qualifiers if they exist.

The following shows two such Oracle Rdb Release 7.2.5.0 RMU/SHOW STATISTICS Configuration File problems. In Oracle Rdb Release 7.2.5.0, when the RMU/SHOW STATISTICS command is invoked with a Configuration File, CONFIGURE.CFG, which specifies that a stall log file, a timeout log file, and a deadlock log file are to be created, an access violation occurs, the command is aborted and no log files are created. In the second example, when the RMU/SHOW STATISTICS command is executed a second time and the command "!A" is executed specifying that the current RMU/SHOW STATISTICS configuration is to be saved to the Configuration File, CONFIGURE2.CFG, another access violation occurs and the command is also aborted.

```
$ RMU/SHOW VERSION
Executing RMU for Oracle Rdb V7.2−500 on OpenVMS Alpha V8.3−V84
$ CREATE CONFIGURE.CFG
STALL_LOG = "a.log";
TIMEOUT_LOG = "b.log";
DEADLOCK_LOG = "c.log";
$ RMU /SHOW STATISTICS /INTERACTIVE MF_PERSONNEL −
        /CONFIGURE=CONFIGURE.CFG
%SYSTEM−F−ACCVIO, access violation, reason mask=00,
 virtual address=000000000000001D, PC=0000000000342A80, PS=0000001B
%RMU−F−FATALOSI, Fatal error from the Operating System Interface.
%RMU−I−BUGCHKDMP, generating bugcheck dump file DEVICE:[DIRECTORY]RMUBUGCHK.DMP;
%RMU−F−FTL_SHOW, Fatal error for SHOW operation at 18−JUL−2011 16:21:37.94
$ DIR *.LOG

%DIRECT−W−NOFILES, no files found
$ RMU/SHOW STATISTICS/INTERACTIVE MF_PERSONNEL
!
A
CONFIGURE2.CFG
%SYSTEM−F−ACCVIO, access violation, reason mask=04,
 virtual address=000000007AA52000, PC=FFFFFFFF81111328, PS=0000001B

  Improperly handled condition, image exit forced by last chance handler.
    Signal arguments:   Number = 0000000000000005
                        Name   = 000000000000000C
                                 0000000000000004
                                 000000007AA52000
                                 FFFFFFFF81111328
                                 000000000000001B

    Register dump:
```

```
    R0  = 000000007AA4B913  R1  = FFFFFFFF813E4E2C  R2  = 000000007BE78408
    R3  = 000000007AA4B8F0  R4  = 0000000000000023  R5  = 0000000022203D20
    R6  = 00000000000E0000  R7  = 0000000000003B22  R8  = 0000000001000000
    R9  = 00000000010E00FF  R10 = 00000000000F8B18  R11 = 00000000001DA1E8
    R12 = 00000000056EC0D4  R13 = 0000000001000000  R14 = 0000000000136498
    R15 = 00000000001CF610  R16 = 000000007AA520D8  R17 = 00000000434EDFF3
    R18 = 0000000000000000  R19 = 0000000000000000  R20 = 0000000000000003
    R21 = 000000007AA4B8F0  R22 = 0000000000000000  R23 = 000000007AA4B7B0
    R24 = 00000000000183B2  R25 = 0000000000000003  R26 = FFFFFFFF813687C4
    R27 = 0000000000000000  R28 = 000000007AA520D8  R29 = 000000007BE6BC00
    SP  = 000000007AA4B7B0  PC  = FFFFFFFF81111328  PS  = 300000000000001B
```

The following example shows that the two Oracle Rdb Release 7.2.5.0 RMU/SHOW STATISTICS Configuration File problems described in the previous example have been fixed. In the first command, the three log files specified by the Configutation File are created and then written to during the RMU/SHOW STATISTICS session. In the second command, a new Configuration File is created containing the current RMU/SHOW STATISTICS configuration.

```
$ CREATE CONFIGURE.CFG
STALL_LOG = "a.log";
TIMEOUT_LOG = "b.log";
DEADLOCK_LOG = "c.log";
$ RMU /SHOW STATISTICS /INTERACTIVE MF_PERSONNEL -
        /CONFIGURE=CONFIGURE.CFG
%RMU-I-LOGCREOUT, created output file DEVICE:[DIRECTORY]A.LOG;1
%RMU-I-LOGCREOUT, created output file DEVICE:[DIRECTORY]B.LOG;1
%RMU-I-LOGCREOUT, created output file DEVICE:[DIRECTORY]C.LOG;1
$ DIR *.LOG

Directory DEVICE:[DIRECTORY]

A.LOG;1             B.LOG;1             C.LOG;1

Total of 3 files.
$ RMU/SHOW STATISTICS/INTERACTIVE MF_PERSONNEL
!
A
CONFIGURE2.CFG

$ DIR CONFIGURE2.CFG

Directory DEVICE:[DIRECTORY]

CONFIGURE2.CFG;1

Total of 1 file.
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.5.2 RMU/SHOW STATISTICS Release 7.2.5.0 Hot Row Information Screen %SYSTEM-F-ACCVIO

The Oracle Rdb RMU/SHOW STATISTICS Row Cache "Hot Row Information" display screen access violates in Oracle Rdb Release 7.2.5.0. This problem does not affect other RMU/SHOW STATISTICS functions. There is no workaround for this problem.

The following shows the problem. In Oracle Rdb Release 7.2.5.0, when the RMU/SHOW STATISTICS command Row Cache "Hot Row Information" display screen is invoked, a %SYSTEM−F−ACCVIO access violation occurs and an RMUBUGCHK.DMP is created with the stack trace displayed.

```
$ RMU/SHOW VERSION
Executing RMU for Oracle Rdb V7.2−500 on OpenVMS Alpha V8.3−V84
$ RMU/SHOW STATISTICS FOO.RDB

Node: NODE01 (1/1/1)    Oracle Rdb V7.2−500 Perf. Monitor 9−AUG−2011 11:27:09.19
Rate: 3.00 Seconds       Row Cache Overview (Unsorted)      Elapsed: 00:01:36.26
Page: 1 of 1          DEVICE:[DIRECTORY]FOO.RDB;1           Mode: Online

Cache.Name.............  #Searches  Hit% Full%  #Inserts #Wrap    #Slots   Len
RDB$SYSTEM_CACHE                  0   0.0   0.0         0     0       500  1000


%SYSTEM−F−ACCVIO, access violation, reason mask=04,
 virtual address=FFFFFFFF800000F0, PC=000000000072F834, PS=0000001B
%RMU−F−FATALOSI, Fatal error from the Operating System Interface.
%RMU−I−BUGCHKDMP, generating bugcheck dump file
DEVICE:[DIRECTORY]RMUBUGCHK.DMP;
%RMU−F−FTL_SHOW, Fatal error for SHOW operation at  9−AUG−2011
11:27:33.56
$ type DEVICE:[DIRECTORY]RMUBUGCHK.DMP

This file was generated by Oracle Rdb V7.2−500 upon detection of a
fatal, unexpected, error.  Please return this file, the query or
program that produced the bugcheck, the database, monitor log, and any
other pertinent information to your Oracle support representative for
assistance.

================================================================================
        Stack Dump Summary
================================================================================

***** Exception at 000000000072F834 : RMU72\KUTDIS$SETUP_RCD_ACTRTN + 000001F4
%SYSTEM−F−ACCVIO, access violation, reason mask=04,
 virtual address=FFFFFFFF800000F0, PC=000000000072F834, PS=0000001B
Saved PC = 000000000066F4C4 : RMU72\KUTDIS$SETUP_COMMON + 00000144
Saved PC = 000000000065B184 : RMU72\KUTDIS$BATCH_SETUP_COMMON + 00000084
Saved PC = 00000000006672A4 : RMU72\KUTDIS$MAIN_MENU_ACTIONS + 000002B4
Saved PC = 000000000065E3E4 : RMU72\KUTDIS$DO_UNSOLICITED_INPUT + 000002B4
Saved PC = 00000000006E3308 : RMU72\KUTDIS$UNSOLICITED_INPUT_ASTX + 000000D8
Saved PC = 000000000065D694 : RMU72\KUTDIS$DISPATCH + 000000A4
Saved PC = 0000000000656344 : RMU72\KUT$DISPLAY + 00001D04
Saved PC = 00000000005B2E78 : RMU72\RMU$DISPLAY + 000036B8
Saved PC = 00000000001FF50C : RMU72\RMU$SHOW + 00000ECC
Saved PC = 00000000003DE45C : RMU72\RMU_DISPATCH + 000013BC
Saved PC = 00000000003DCC6C : RMU72\RMU_STARTUP + 000004FC
Saved PC = 00000000001F0F24 : RMU72\RMU$MAIN + 00000034
Saved PC = 000000007AECF5D8 : Image DCL + 000935D8
```

The following example shows that this problem has been fixed and the RMU/SHOW STATISTICS Row Cache "Hot Row Information" screen is now displayed correctly.

```
$ RMU/SHOW STATISCTICS FOO.RDB

Node: NODE01 (1/1/1)    Oracle Rdb V7.2−510 Perf. Monitor  9−AUG−2011
 11:20:20.64
Rate: 3.00 Seconds            Hot Row Information           Elapsed: 00:04:34.18
```

```
Page:  1 of 16        DEVICE:[DIRECTORY]FOO.RDB;1          Mode: Online

                     For Cache: RDB$SYSTEM_CACHE (unsorted)
Area:Page:Ln #Users State Length SlotNo Area:Page:Ln #Users State Length SlotNo
Empty             0            0      0 Empty             0            0     16
Empty             0            0      1 Empty             0            0     17
Empty             0            0      2 Empty             0            0     18
Empty             0            0      3 Empty             0            0     19
Empty             0            0      4 Empty             0            0     20
Empty             0            0      5 Empty             0            0     21
Empty             0            0      6 Empty             0            0     22
Empty             0            0      7 Empty             0            0     23
Empty             0            0      8 Empty             0            0     24
Empty             0            0      9 Empty             0            0     25
Empty             0            0     10 Empty             0            0     26
Empty             0            0     11 Empty             0            0     27
Empty             0            0     12 Empty             0            0     28
Empty             0            0     13 Empty             0            0     29
Empty             0            0     14 Empty             0            0     30
Empty             0            0     15 Empty             0            0     31

Config Exit Help Menu >next_page <prev_page ]next_obj [prev_obj Options Set_rate
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1.

2.5.3 Unexpected Failure in COSI_MEM_FREE_VMLIST When Using RMU Show Statistics

Bug 9853784

In prior versions of Oracle Rdb, the RMU Show Statistics command may abort with a bugcheck similar to the following footprint.

- Itanium OpenVMS 8.3−1H1
- Oracle Rdb Server 7.2.4.1.0
- Got a RMUBUGCHK.DMP
- SYSTEM−F−ACCVIO, access violation, virtual address=0000000034383030
- Exception occurred at RMU72\COSI_MEM_FREE_VMLIST + 00000132
- Called from RMU72\KUTDIS$TOOLS_CONTROL + 00010170
- Called from RMU72\KUTDIS$TOOLS_MENU + 0000DBC0
- Called from RMU72\KUTDIS$MAIN_MENU_ACTIONS + 000032D0
- Running image RMU72.EXE
- Command line was: RMUI/SHOW STAT MF_PERSONNEL

This occurs when the list of process ID's listed by SHOW STATISTICS exceeds an internal buffer (approximately 144 entries).

For instance, when entering this sequence of commands on the problem system, RMU wanted to display a list of 200 PID values.

- !
- R (<<more>>)
- J (Process monitoring)
- B (Activate specific eligible process)

- When prompted answer with "?" to get a full list of process ids.

Note

Actual menu selections may differ depending on the Rdb version being used.

This problem has been corrected in Oracle Rdb Release 7.2.5.1. RMU SHOW STATS now dynamically allocates the list to accommodate a virtually unlimited list of process ID's.

2.5.4 Invalid Average Transaction Duration Value Displayed When Using RMU Show Statistics

Bug 13017360

With Oracle Rdb Release 7.2.5, the Oracle Rdb RMU Show Statistics "Transaction Duration" screen displayed an incorrect average transaction duration value.

The command to see that statistics screen is:

```
$ RMU/SHOW STATISTICS MF_PERSONNEL/SCREEN="Transaction Duration (Total)"
```

In most cases, RMU SHOW STATS would only display "<−avg=0.000000". The value displayed was 1/10,000th of the true value, leading to a much smaller number than the actual value.

This problem has been corrected in Oracle Rdb Release 7.2.5.1. RMU SHOW STATS now shows the correct statistics for the average transaction duration.

2.5.5 RMU Show Statistics Sometimes Bugchecks When Using Process Monitoring

Bug 13070947

The Oracle Rdb RMU Show Statistics command "Process monitoring" option would sometimes bugcheck. The following sequence of commands, while in an RMU/SHOW STATISTICS window, would sometimes result in a bugcheck.

```
 ! (to get to the Select Tool)
 R. <<more>>
 J. Process monitoring
 B. Activate specific eligible process
 Enter process ID to "activate" (or "?"): (enter id here)

 ! (to get to the Select Tool)
 J. Process monitoring
 A. Select activated process to monitor
 Process detached; Continue reviewing? ("Yes" to continue, "RETURN" to cancel)
```

The bugcheck looked like the following:

```
***** Exception at FFFFFFFF84230400 : Image LIBOTS + 00002400
%SYSTEM−F−ACCVIO, access violation, reason mask=00,
```

```
virtual address=0000000003A64000, PC=FFFFFFFF84230400, PS=0000001B
Saved PC = 000000008089AB10 : RMU72\KUTDIS$TOOLS_CONTROL + 0000EE80
Saved PC = 00000000809855A0 : RMU72\KUTDIS$TOOLS_MENU + 0000DBC0
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1. RMU will no longer bugcheck and will show the requested statistics.

2.5.6 RMU Show Statistics Sometimes Bugchecks on Row Cache Information Screen

Bug 13558012

The Oracle Rdb RMU Show Statistics command "Hot Row Information" option would sometimes bugcheck. The following sequence of commands, while in an RMU/SHOW STATISTICS window, would sometimes result in a bugcheck when monitoring row cache.

```
 M (Menu)
 X (Row Cache Information)
 B (Hot Row Information)
 Select one cache
```

The bugcheck had the following footprint:

```
SYSTEM-F-ACCVIO, access violation, virtual address=FFFFFFFF817B80B0
Exception occurred at RMU72\KUTDIS$SETUP_RCD_ACTRTN + 00000460
```

This problem has been corrected in Oracle Rdb Release 7.2.5.1. RMU will no longer bugcheck and will show the requested statistics.

Chapter 3
Software Errors Fixed in Oracle Rdb Release 7.2.5

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.5.

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 Server Process Name Format Changed

When starting server processes (such as database recovery processes), previous releases of Oracle Rdb would always start creating processes with a name starting with a number one (such as RDM_RDB72_1). In some cases, when starting a large number of processes or when multiple databases were opened on a system, duplicate names would be created and processes would have to be re−started with a different process name. This resulted in additional system resources being consumed.

This problem has been reduced in Oracle Rdb Release 7.2.5. Server processes are created with much more unique names.

3.1.2 Drop Storage Area Cascade Failed With Lock On Unrelated Area

Bug 7496558

When dropping a storage area using the CASCADE option, a lock on an unrelated area could have caused the drop to fail.

For example, if an RMU/UNLOAD was running which referenced storage areas unrelated to the target area of the ALTER DATABASE ... DROP STORAGE AREA ... CASCADE statement, a LOCK_CONFLICT error was reported.

```
SQL> ALTER DATABASE FILENAME DDLLOCK DROP STORAGE AREA T5 CASCADE;
%RDB−E−LOCK_CONFLICT, request failed due to locked resource
−RDMS−F−LCKCNFLCT, lock conflict on client 'DDL' 4C444400000055
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.1.3 Temporary File Names

Oracle Rdb generates random file names for various temporary functions (such as AIJ recovery work files). In rare cases, the file names would not be unique in a cluster and could potentially cause a conflict.

This problem has been corrected in Oracle Rdb Release 7.2.5. Oracle Rdb now generates file names that are unique within a cluster.

3.1.4 Incorrect Storage Area Selected In Cluster

Bug 9629294

In an environment where a database is opened on multiple nodes of a cluster, it is possible that certain combinations of dropping and adding storage areas online with storage area names being re−used may result

in storage map creation selecting an incorrect area.

The following command sequence on two nodes of a cluster shows one possible case where a table is incorrectly mapped to a storage area not specified. Note that the final "RMU/ANALYZE/AREA=A2 TESTDB" command output shows both tables T1 and T2 being unexpectedly stored in area A2 even though the storage map has specified area A1 for table T1.

```
        NODEA                           NODEB
------------------------------ ------------------------------------
$ SQL$
 CREATE DATABASE FILE TESTDB
 OPEN IS MANUAL
 RESERVE 10 STORAGE AREAS
 CREATE STORAGE AREA DUMMY;
 EXIT
$ RMU /OPEN TESTDB

                                $ RMU /OPEN TESTDB
                                $ SQL$
                                 ALTER DATABASE FILE TESTDB
                                 ADD STORAGE AREA A1
                                 ADD STORAGE AREA A2;
                                 ATTACH 'FILE TESTDB';
                                 CREATE TABLE T1 ( I1 INT );
                                 CREATE STORAGE MAP M1
                                        FOR T1 STORE IN A1;
                                 CREATE TABLE T2 ( I1 INT );
                                 CREATE STORAGE MAP M2
                                        FOR T2 STORE IN A2;
                                 COMMIT;
                                 EXIT
                                $ RMU /CLOSE TESTDB

$ RMU/CLOSE TESTDB

                                $ RMU/OPEN TESTDB

$ RMU /OPEN TESTDB
$ SQL$
 ATTACH 'FILE TESTDB';
 DROP TABLE T1;
 DROP TABLE T2;
 COMMIT;

                                $ SQL$
                                 ALTER DATABASE FILE TESTDB
                                        DROP STORAGE AREA A1
                                        DROP STORAGE AREA A2;
                                 ALTER DATABASE FILE TESTDB
                                        ADD STORAGE AREA A2;
                                 ALTER DATABASE FILE TESTDB
                                        ADD STORAGE AREA A1;
                                 EXIT

 CREATE TABLE T1 ( I1 INT );
 CREATE STORAGE MAP M1
        FOR T1 STORE IN A1;
 CREATE TABLE T2 ( I1 INT );
 CREATE STORAGE MAP M2
        FOR T2 STORE IN A2;
 COMMIT;
```

```
 EXIT

$ RMU/ANALYZE/AREA=A2 TESTDB
... Both tables are stored in A2
```

This problem was caused by a failure to invalidate a cache of storage area names during a search for a matching name. Because the cache was stale, an incorrect storage area slot was selected.

This problem has been corrected in Oracle Rdb Release 7.2.5. The internal cache of storage area names is now correctly synchronized in a cluster environment.

3.1.5 Unexpected SYSTEM−F−VA_NOTPAGALGN Error With Global Buffers and Reserved Memory Registry

Bug 8204438

In some cases, when using database global buffers along with a resident database global section along with the global section being allocated via the OpenVMS Reserved Memory Registry, opening the database may fail with the error SYSTEM−F−VA_NOTPAGALGN as in the following example:

```
$ RMU/OPEN FOO/GLOBAL_BUFFERS=TOTAL=1231
%RDMS−F−CANTOPENDB, database could not be opened as requested
−RDMS−F−CANTCREGBL, error creating and mapping database global section
−SYSTEM−F−VA_NOTPAGALGN, specified virtual address is not CPU−specific page
aligned
```

This problem has been corrected in Oracle Rdb Release 7.2.5. As a workaround if this problem is encountered, the memory reservation for the database global section can be removed as in the following example:

```
$ MCR SYSMAN RESERVED_MEMORY FREE RDM72N$1$DGA242296000400000000000
%SMI−S−RMRFREPAG, pages successfully freed from reservation
$ RMU/OPEN FOO/GLOBAL_BUFFERS=TOTAL=1231
```

3.1.6 Unexpected Bugcheck at RDMS$$PARSE_INTCOM_BUFFER Which Reports "Obsolete Version of Database"

Bugs 460614, 3314889, 3655192, 3658460, 6988338, 8271388, 8616430, 8785676, 9206054 and 9887582

In prior releases of Oracle Rdb, it was possible in rare circumstances to have a bugcheck generated similar to that shown below. This problem occurred during database attach and was due to a timing issue related to asynchonous database events.

- Itanium OpenVMS 8.3−1H1
- Oracle Rdb Server 7.2.3.1.0
- Got a RDSBUGCHK.DMP
- RDB−F−WRONGRDB, RDB$SHARE image is wrong
- RDMS−F−OBSVER, obsolete version of database

- Exception occurred at RDMSHRP72\RDMS$$PARSE_INTCOM_BUFFER + 00000740
- Called from RDMSHRP72\KODSTREAM$JACKET + 00000100
- Called from symbol not found
- Called from RDMSHRP72\KOD$SETSTK_AND_CONTINUE + 00000180

This problem has been corrected in Oracle Rdb Release 7.2.4.2 for the ATTACH, CONNECT and DECLARE ALIAS statements. This release, 7.2.5, also corrects other areas for DISCONNECT, SET SESSION AUTHORIZATION, DROP DATABASE, and ALTER DATABASE.

3.1.7 RDBPRE Precompiler RUNTIMSTK Informational Message From MACRO Compiler

In some cases, the RDBPRE Precompiler may create code that causes the MACRO−32 Compiler for OpenVMS I64 and the MACRO−32 Compiler for OpenVMS Alpha to display an informational message similar to "%IMAC−I−RUNTIMSTK, run time stack differences prevent accurate stack tracing". This informational message can be safely ignored.

Consider the following example program:

```
PROGRAM−ID. x.
DATA DIVISION.
WORKING−STORAGE SECTION.

&RDB& INVOKE DATABASE EXTERNAL d1 = FILENAME "PERSONNEL"
&RDB& INVOKE DATABASE EXTERNAL d2 = FILENAME "PERSONNEL"

PROCEDURE DIVISION.
01−INITIALIZE−AND−PROCESS.
&RDB&   FINISH
        DISPLAY "Hello World".
        EXIT PROGRAM.

END PROGRAM x.
```

During compilation, the "RUNTIMSTK" informational message is displayed:

```
$ RDBPRE /COBOL X.RCO

3$:
^
%IMAC−I−RUNTIMSTK, run time stack differences prevent accurate
stack tracing at line number 214 in file DUA0:[RDB73]X.MAR;1
```

This issue has been corrected in Oracle Rdb Release 7.2.5. The generated code for the FINISH section has been modified to avoid the "RUNTIMSTK" informational message.

3.1.8 Bugcheck At RUJUTL$ROLLBACK_LOOP

Bug 9856675

In very rare cases, it is possible for a rollback operation (either explicit or implicit) to fail with a bugcheck due to entries being unable to be "undone" on a database page due to an unexpected lack of "locked" space. The sequence of events is complex and requires a specific ordering of operations and accumulation of locked and

free space on a database page among several processes.

The bugcheck "footprint" will be similar to the following:

```
Exception occurred at RDMSHRP72\RUJUTL$ROLLBACK_LOOP + 000010A1
Called from RDMSHRP72\RUJ$ROLLBACK + 000000F0
Called from RDMSHRP72\KOD$ROLLBACK + 000007A0
Called from RDMSHRP72\RDMS$$INT_ROLLBACK_TRANSACTION + 00001140
Called from RDMSHRP72\RDMS$TOP_ROLLBACK_TRANSACTION + 00000A90
```

Analysis of the bugcheck dump will indicate one or more entries on the "FBIJBL" queue similar to the following:

```
FBIJBL @1C3109C0:         QUE = 16E5B0F8:16E5B0F8
    +-------------------------------------------------------+
    | This JFA 0            Record sequence number 640      |
    | Prior JFA 94096       Previous TSN was 0:3390022611    |
    | Modified segment 911:11828:16 with length of 64 bytes  |
    +-------------------------------------------------------+
```

The cause of the problem was related to an incorrect synchronization between processes manipulating the locked and free space while adding lines to the page.

This problem has been corrected in Oracle Rdb Release 7.2.5. Oracle recommends that all Rdb installations upgrade to at least Oracle Rdb Release 7.2.5 to implement the correction.

3.1.9 ALTER TABLE Fails With Constraint Violation

Bug 4904254

A customer reported that RMU/VERIFY/ALL reported incorrect constraint failure.

```
%RMU-W-CONSTFAIL, Verification  of constraint
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-INTEG_FAIL, violation of constraint CODE_NOT_IN_T2 caused operation
to fail
```

The constraint was checked in SQL and showed no violations.

The following ALTER TABLE statement is the equivalent SQL which shows the constraint failure:

```
alter table T1 add constraint
 constraint CODE_NOT_IN_T2
 check
 (exists
 (select pcode from T2 a
  where a.pcode = T1.pcode and
        a.ptype in
        ('BOX','BUL','COL','ROL','TPP','TPN','CON','SHE')))
 deferrable;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-INTEG_FAIL, violation of constraint CODE_NOT_IN_T2 caused operation
to fail
-RDB-F-ON_DB, on database <directory_spec>TESTDB.RDB;1
rollback;
```

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

1. The main query is an ALTER TABLE statement on a table to add a constraint
2. The CHECK contains an EXISTS statement of SELECT query with IN clause and a join predicate

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.1.10 Increased Default for RDMS$BIND_WORK_VM and Relocation of Related VM Buffer to P2 Virtual Address Space

When executing certain kinds of queries that yield large record streams, the Oracle Rdb optimizer may have to create an intermediate table to store the results of a subquery. Oracle Rdb stores these results in sorted order for further execution in join operations.

The RDMS$BIND_WORK_VM logical name can reduce the overhead of disk I/O operations for matching operations that utilize the internal intermediate table. This logical name lets you specify the amount of virtual memory (VM) in bytes that will be allocated to your process for use as a buffer for the subquery results. Once the allocation is exhausted, additional data values are written to a temporary file on disk.

In prior versions of Oracle Rdb, the buffers for these subquery results were allocated in 32−bit process (P0) virtual address space. For some complex queries, the 1GB size of P0 space drastically limited the viable size of the internal buffers where larger buffers would otherwise increase performance by avoiding disk file IO.

This problem has been corrected in Oracle Rdb Release 7.2.5. The internal buffers have been moved to 64−bit process (P2) virtual address space in order to both reduce use of 32−bit process (P0) virtual address space and to permit much larger buffers to be utilized. In addition, the default value for the RDMS$BIND_WORK_VM logical name has been increased from 10,000 bytes to 100,000 bytes. The maximum value is 2,147,483,647 (2GB).

3.1.11 Full Outer Join Query Returns Wrong Column Values When Outer Table is Empty

Bug 9975516

In prior releases of Oracle Rdb, the following example returns wrong column values when the outer table is empty:

```
create table ta (fx char(3), fy char(1));
create table tb (fx char(3), fy char(1));

create unique index ia on ta (fx);
create unique index ib on tb (fx);

! insert one row in TA
insert into ta values ( 'AAA', '1');

! insert two rows in TB
insert into tb values ( 'ABC', '1');
insert into tb values ( 'BBB', '1');
```

```
! the full outer join returns correctly
select * from ta a full outer join tb b on a.fx = b.fx;
 A.FX   A.FY   B.FX   B.FY
 AAA    1      NULL   NULL
 NULL   NULL   ABC    1
 NULL   NULL   BBB    1
3 rows selected

!If all the rows in table "ta" are deleted, the same query returns the correct
!number of rows but wrong column values for the inner table "tb":

delete from ta;
1 row deleted

select * from ta a full outer join tb b
on a.fx = b.fx;
 A.FX   A.FY   B.FX   B.FY
 NULL   NULL   ...    .
 NULL   NULL   ...    .
2 rows selected

!We would expect to see the following correct result:
 A.FX   A.FY   B.FX   B.FY
 NULL   NULL   ABC    1
 NULL   NULL   BBB    1
2 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.1.12 Reduction in Use of Rdb Executive Sort P0 Address Space

Previously, large data structures used by internal SORT code within Rdb were allocated in program (P0) virtual memory address space. In some cases, these data structures could consume considerable space and could lead to exceeding the capacity of P0 address space.

The impact of this situation has been reduced in Oracle Rdb Release 7.2.5. Several of the large data structures used by internal SORT code within Rdb have been moved to 64−bit P2 virtual memory address space.

3.1.13 Attaching to Rdb at Remote Site Stalls

Bug 9263939

Sometimes when attempting to attach to a remote Rdb database, the local process would enter a wait state and stall. This would most likely occur when there were other processes also attempting remote attaches.

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.1.14 Increased Default Use of "Quick Sort"

The default size of an internal "quick sort" buffer has been increased from 20,000 bytes to 409,600 bytes and the default record limit for an internal "quick sort" has been increased from 63 to 5000. These changes should

provide for improved performance of sort operations for a larger set of cases.

Note that this internal buffer is located in P2 virtual address space so the increased size should not impact use of P0 virtual address space. It is possible that the increased default size might require additional process working set size in order to avoid excessive page faulting.

The logical name RDMS$BIND_MAX_QSORT_COUNT is used to restrict the number of rows that can be stored within the "quick sort" buffer. If required to revert to the prior maximum record count, this logical name can be defined as follows:

- DEFINE RDMS$BIND_MAX_QSORT_COUNT 63

Duplicate Handling During Sort Operations

The handling of row items duplicate key values during any sort operation is undefined in terms of the returned row order. Consider a query similar to "SELECT A,B FROM T ORDER BY A". If there are duplicate values for colum A, the order of output for column B is undefined and may potentially change from one execution of the query to another. If a specific order of values for column B is required, it must be explicitly specified in the ORDER BY clause.

3.1.15 Bugcheck While In PSII2INSERTDUPBBC

In unusual cases, when inserting duplicate entries into a sorted ranked index, it was possible for Rdb to bugcheck with an exception "footprint" similar to the following:

```
***** Exception at FFFFFFFF80002840 : symbol not found
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000010048000, PC=FFFFFFFF80002840, PS=00000009
Saved PC = FFFFFFFF856874D0 : RDMSHRP72\PSII2INSERTDUPBBC + 00003230
Saved PC = FFFFFFFF8567F9E0 : RDMSHRP72\PSII2INSERTBOTTOM + 00000B80
Saved PC = FFFFFFFF85664900 : RDMSHRP72\PSII2INSERTT + 00000400
Saved PC = FFFFFFFF85664E00 : RDMSHRP72\PSII2INSERTT + 00000900
Saved PC = FFFFFFFF85664E00 : RDMSHRP72\PSII2INSERTT + 00000900
Saved PC = FFFFFFFF85664E00 : RDMSHRP72\PSII2INSERTT + 00000900
Saved PC = FFFFFFFF85664E00 : RDMSHRP72\PSII2INSERTT + 00000900
Saved PC = FFFFFFFF85664E00 : RDMSHRP72\PSII2INSERTT + 00000900
Saved PC = FFFFFFFF856676F0 : RDMSHRP72\PSII2INSERTTREE + 00000450
Saved PC = FFFFFFFF85C86E30 : RDMSHRP72\RDMS$$KOD_INSERT_TREE + 00006AB0
Saved PC = FFFFFFFF85C04160 : RDMSHRP72\RDMS$$EXE_ACTION + 00001480
Saved PC = FFFFFFFF85D00E40 : RDMSHRP72\RDMS$$C_EXE_ACTION + 00000080
Saved PC = FFFFFFFF855C9DA0 : RDMSHRP72\RDMS_EXE_INTERP + 0000FD50
Saved PC = FFFFFFFF85C61690 : RDMSHRP72\RDMS$TOP_RECEIVE_BUFFER + 00001FC0
Saved PC = FFFFFFFF85C56340 : RDMSHRP72\RDMS$TOP_START_AND_SEND + 00001D80
Saved PC = FFFFFFFF866FC1A0 : RDMSHRP72\AMAC$EMUL_CMPC5 + 00002040
Saved PC = FFFFFFFF86461E40 : RDMSHRP72\KODSTREAM$JACKET + 00000130
```

In some cases, this problem may have been caused by an internal data structure being overwritten, leading to an incorrect length being calculated. It was possible for other memory structures to be compromised as well, leading to various different symptoms.

This problem has been corrected in Oracle Rdb Release 7.2.5. The data structure length is now correctly

calculated. All customers utilizing SORTED RANKED indexes are encouraged to upgrade to this release.

3.1.16 Divide Operator Now Returns DOUBLE PRECISION Results Rather than REAL

With this release of Oracle Rdb, all division operations (including the AVG statistical function) will produce DOUBLE PRECISION results. In prior releases, divide operations that involved small numeric values (TINYINT, SMALLINT, and short integer literals) resulted in REAL results (single precision) with some loss of accuracy.

This will affect AVG or expressions that contain division (/) operators. The result might be visible as a change in type for COMPUTED BY columns, AUTOMATIC AS columns, and view select expressions. Existing applications which expect to receive REAL results will cause Oracle Rdb to implicitly convert from DOUBLE PRECISION to REAL. Applications that use Dynamic SQL should ensure that the handling of floating types is consistent with either REAL or DOUBLE PRECISION results.

In addition, on Itanium systems, some arithmetic opertions that result in REAL results (single precision floating point) previously were performed in single precision IEEE S floating point format. These arithmetic operations are now performed in double precision IEEE T floating point format and the results are converted to REAL (single precision floating point) format. This may result in slightly greater precision in some cases.

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.1.17 Unexpected Results From IN Clause on a Subselect Containing FETCH FIRST or LIMIT TO

Bugs 10244544 and 9500560

In prior versions of Oracle Rdb, the "Index counts lookup" optimization could be incorrectly applied to a query using a SORTED RANKED index that also included a LIMIT TO (or FETCH FIRST) clause. The result was extra rows being returned from the query.

The following example shows this problem with an IN clause referencing a subquery containing a FETCH FIRST ROWS ONLY clause. This clause is equivalent to the Rdb SQL LIMIT TO 1 ROWS clause.

```
SQL> select a
cont>  from tt11
cont>  where a in (select a
cont>        from tt11
cont>        order by a desc
cont>        fetch first row only);
Tables:
  0 = TT11
  1 = TT11
Cross block of 2 entries  Q1
  Cross block entry 1
    Index only retrieval of relation 0:TT11
      Index name  TT11_INDEX [0:0]
  Cross block entry 2
    Conjunct: <agg0> <> 0
    Aggregate-F1: 0:COUNT-ANY (<subselect>) Q2
    Conjunct: 0.A = 1.A
```

```
    Firstn: 1
    Index only retrieval of relation 1:TT11
      Index name  TT11_INDEX [0:0]  Reverse Scan  Index counts lookup
           A
         164
         164
...
         471
         471
274 rows selected
SQL>
```

The correct results should be limited to the rows matching only the maximum values (selected by sorting the values in descending order) for TT11 column A.

```
SQL> select a
cont>  from tt11
cont>  where a = (select a
cont>       from tt11
cont>       order by a desc
cont>       fetch first row only);
Tables:
  0 = TT11
  1 = TT11
Cross block of 2 entries  Q1
  Cross block entry 1
    Aggregate: 0:VIA (1.A) Q2
    Firstn: 1
    Index only retrieval of relation 1:TT11
      Index name  TT11_INDEX [0:0]   Reverse Scan
  Cross block entry 2
    Index only retrieval of relation 0:TT11
      Index name  TT11_INDEX [1:1]
        Keys: 0.A = <agg0>
           A
         471
         471
         471
3 rows selected
SQL>
```

The workaround for this problem is to define RDMS$SET_FLAGS as "NOCOUNT_SCAN" or use SET FLAGS 'NOCOUNT_SCAN' in SQL.

This problem has been corrected in Oracle Rdb Release 7.2.5. The Rdb optimizer now detects this case and automatically disables the "Index counts lookup" optimization.

3.1.18 Translation From HEX Character Set is Incorrect

Bug 10265503

A problem with character set translation, introduced in V7.2 Oracle Rdb, will prevent the correct translation of literals, variables and columns if the character set of the source object is HEX.

The following example shows this problem.

```
SQL> select translate (_hex'4142' using rdb$isolatin1) from rdb$database;
```

```
 3431
1 row selected
SQL>
```

The correct results should be the translation of the hexadecimal value to the appropriate characters within the destination character set.

```
SQL> select translate (_hex'4142' using rdb$isolatin1) from rdb$database;

 AB
1 row selected
SQL>
```

A workaround for this problem, if the source is a literal, is to use the hexadecimal literal specifier as in the following example.

```
SQL> select x'4142'  from rdb$database;

 AB
1 row selected
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.5. Rdb now carries out the correct translation.

3.1.19 Nested Query With Left Outer Join and GROUP BY Bugchecks During Query Compilation

Bug 10266984

In prior releases of Oracle Rdb, the following query, nested with LEFT OUTER JOIN and GROUP BY, bugchecks during query compilation.

```
SELECT
  dt1.inv_date,
  dt1.method,
  sum(dt1.cnt)
FROM    ! derived table:dt1
 (SELECT
    dt2.inv_id,
    dt2.inv_date,
    t4.method,
    dt2.cnt
  FROM  ! derived table:dt2
   (SELECT
        t1.inv_id,
        t1.inv_date,
        t2.bid,
        t3.sid,
        sum(t2.quantity)
    FROM
        HEADER t1 JOIN
        DETAIL t2
            on (t2.inv_id=t1.inv_id)
        LEFT JOIN SPLIT t3
            on (t3.inv_id = t1.inv_id and
                t3.detail_glue = t2.detail_glue)
```

```
    WHERE
          t1.inv_date>'01-MAR-2010' and
          t1.cancel = 'F' and
          t1.form = 'F' and
          t2.quantity > 0
    GROUP BY
        t1.inv_id,
        t1.inv_date,
        t2.bid,
        t3.sid
     ) dt2        ! Derived table
        (inv_id,
         inv_date,
         bid,
         sid,
         cnt)
    JOIN SO_INVOICE_BOX_REC t4
         on (t4.inv_id = dt2.inv_id and
             t4.bid = dt2.bid )
    WHERE t4.method <> 'OTHR' or
          t4.cid <> 'OTHR' or
          t4.weight > 1
   ) dt1 (inv_id,    ! Derived table
        inv_date,
        method,
        cnt)
  GROUP BY
    dt1.inv_date,
    dt1.method
    limit 5;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DISK:[DIRECTORY]RDSBUGCHK.DMP;
%RDB-F-BUG_CHECK, internal consistency check failed
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.1.20 Query With Nested Left Outer Join Bugchecks With Floating Overflow

Bug 10185583

In prior releases of Oracle Rdb, the following query, with nested LEFT OUTER JOIN, bugchecks with floating overflow.

```
SELECT count(*)
FROM TABLE1
LEFT OUTER JOIN TABLE2
ON TABLE2.COL_193= 'C10014' and TABLE2.COL_194= '00'

LEFT OUTER JOIN TABLE3
ON TABLE3.COL_193= 'C10014' and TABLE3.COL_194= '00'

LEFT OUTER JOIN TABLE4
ON TABLE4.COL_193= 'C10014' and TABLE4.COL_194= '00'

LEFT OUTER JOIN
  (select COL_197, COL_198, COL_046 ALIAS014 from TABLE5 where COL_197='C10014'
   and COL_198='00' and COL_039='EIS' and COL_043='REL' and COL_200=1) TBL_TBL008
ON TABLE1.COL_193=TBL_TBL008.COL_197 and TABLE1.COL_194=TBL_TBL008.COL_198
```

```
LEFT OUTER JOIN
  (select COL_197, COL_198, COL_046 ALIAS015 from TABLE5 where COL_197='C10014'
   and COL_198='00' and COL_039='EIS' and COL_043='REL' and COL_200=2) TBL_TBL009
ON TABLE1.COL_193=TBL_TBL009.COL_197 and TABLE1.COL_194=TBL_TBL009.COL_198

LEFT OUTER JOIN
  (select COL_197, COL_198, COL_046 ALIAS010 from TABLE5 where COL_197='C10014'
   and COL_198='00' and COL_039='EIS' and COL_043='GER' and COL_200=1) TBL_TBL004
ON TABLE1.COL_193=TBL_TBL004.COL_197 and TABLE1.COL_194=TBL_TBL004.COL_198

...followed by a series of left outer join here...

LEFT OUTER JOIN
  (select COL_197, COL_198, COL_046 ALIAS007 from TABLE5 where COL_197='C10014'
   and COL_198='00' and COL_039='APA' and COL_043='APA') tbl_ALIAS007
ON TABLE1.COL_193=tbl_ALIAS007.COL_197 and TABLE1.COL_194=tbl_ALIAS007.COL_198

LEFT OUTER JOIN
  (select COL_197, COL_198, COL_046 ALIAS048 from TABLE5 where COL_197='C10014'
   and COL_198='00' and COL_039='TYP' and COL_043='GIN') TBL_TBL033
ON TABLE1.COL_193=TBL_TBL033.COL_197 and TABLE1.COL_194=TBL_TBL033.COL_198

WHERE TABLE1.COL_193= 'C10014'
      AND TABLE1.COL_194= '00'
;
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-SYSTEM-F-HPARITH, high performance arithmetic trap, Imask=00000000, Fmask=00000
001, summary=08, PC=00000000008062F4, PS=0000001B
-SYSTEM-F-FLTOVF, arithmetic trap, floating overflow at PC=00000000008062F4, PS=
0000001B
```

This query works if the SET FLAGS OLD_COST_MODEL is applied.

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.1.21 DBR Process Waiting for RMS Lock While Adding Process Rights

In rare cases, it is possible for a database recovery (DBR) process to be involved in a deadlock involving a lock on the system RIGHTSLIST file.

This problem has been corrected in Oracle Rdb Release 7.2.5. The DBR process does not grant itself additional rights until after it has gained the database freeze lock and notified the monitor that the failed user process may exit.

3.1.22 DBR Bugcheck at RUJUTL$ROLLBACK_LOOP + 00000760

Bug 10296522

Under certain circumstances, it may be possible for a Database Recovery Process (DBR) to fail while trying to recover a user process, even when there is nothing to recover. This failure would generate a RDMDBRBUG.DMP file and cause the database to shutdown. Once the shutdown occurs, a subsequent

recovery operation will be successful and the database will again be accessible.

For this problem to occur, Row Cache must be enabled and an RMU/BACKUP/AFTER must occur between the time that a user process commits one transacation and starts another transaction, which then fails before doing any database updates.

The DBR fails because of a missing checkpoint record.

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.1.23 Rdb Monitor Log File Write Rate Reduced

The Oracle Rdb Monitor process (RDMMON) previously would write to the monitor log file using an IO size of either 127 or 124 disk blocks and would perform a RMS "flush" operation once per minute.

With this release of Oracle Rdb, the Monitor process now anticipates that OpenVMS patch(es) have been installed that support using a RMS Multi Block Count (MBC) parameter larger than 127 blocks. The Oracle Rdb Monitor process will first attempt to use a larger value and if a RMS$_MBC error is returned from the SYS$CONNECT call to the log file, a second attempt is made with a RMS Multi Block Count (MBC) parameter of less than 128.

In addition, the Oracle Rdb Monitor process reduces the frequency of monitor log file "flush" operations to once every ten minutes.

For some sites with significant monitor log file write activity, these changes should serve to reduce physical IO.

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.1.24 Memory Layout Change For Global Section

Bug 11741649

In order to help prevent cases where shared memory is unexpectedly overwritten, the internal layout of certain data structures has been altered to include additional "guard" pages within Rdb's memory management subsystem.

These changes are intended to help detect and prevent unexpected memory write access to the database global section.

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.1.25 CONCAT on Operands of Same Datatype and Same Size Bugchecks

Bug 11867911

In prior releases of Oracle Rdb, the following query with CONCAT bugchecks.

```
SELECT COUNT(*) FROM T1 WHERE
  COL1 = '13900' AND
  COL2
      || COL3
      || COL4
      || CAST(COL5 + 1999 AS CHAR(4))
   >
    '            '
      || '   '
      || '        '
      || CAST(0 + 1999 AS CHAR(4))
   ;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file _[directory]RDSBUGCHK.DMP;
```

The query works if the dialect is set to 'ORACLE LEVEL2', as in the following example.

```
SET DIALECT 'ORACLE LEVEL2';

SELECT COUNT(*) FROM T1 WHERE
  COL1 = '13900' AND
  COL2
      || COL3
      || COL4
      || CAST(COL5 + 1999 AS CHAR(4))
   >
    '            '
      || '   '
      || '        '
      || CAST(0 + 1999 AS CHAR(4))
   ;
Tables:
  0 = T1
Aggregate: 0:COUNT (*) Q2
Leaf#01 BgrOnly 0:T1 Card=244646
  Bool: (0.COL1 = '13900') AND (CONCAT (0.COL2, 0.COL3,
        0.COL4, CAST ((0.COL5 + 1999) AS CHAR(4))) >
        CONCAT ('            ', '   ', '        ', CAST ((0 + 1999) AS CHAR(4)))
        )
  BgrNdx1 T1_NDX1 [1:1] Fan=8
    Keys: 0.COL1 = '13900'
  BgrNdx2 T1_NDX2 [0:0] Fan=14
    Bool: CONCAT (0.COL2, 0.COL3, 0.COL4, CAST ((
          0.COL5 + 1999) AS CHAR(4))) > CONCAT ('            ', '   ',
          '        ', CAST ((0 + 1999) AS CHAR(4)))

        8525
1 row selected
```

The query bugchecks when the left operands of the CONCAT are the same datatype and same size as the right operands.

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.1.26 SQLSRV−E−PWDEXPIRED Error Restored

Bug 11831591

In Oracle SQL/Services releases prior to 7.3.0.3, if a user account's password was expired, an attempt to

connect got the SQLSRV−E−PWDEXPIRED error. When intrusion detection was added in Release 7.3.0.3 of SQL/Services, this error changed to SQLSRV−F−GETACCINF. Therefore, applications were unable to trap expired password errors in order to prompt the user for a new password.

In Oracle SQL/Services Release 7.3.1, the SQLSRV−E−PWDEXPIRED error has been restored and the SQLSRV−F−GETACCINF error will no longer be returned in this case.

This kit provides the RDB$COSIP.EXE image required to support this change to the SQL/Services behavior.

3.1.27 Query Returns Wrong Result and Bugchecks at Exit Using Bitmapped Scan

Bug 11076142

In prior releases of Oracle Rdb, the following query would return wrong results using Bitmapped Scan and would bugcheck at exit time.

```
set flags 'bitmapped_scan';
select distinct DETAIL  from T2 B, T1 M
where B.TYP = M.PID and
      M.NUMTYP = 'T_TYP' ;
Tables:
  0 = T2
  1 = T1
Reduce: 0.DETAIL
Sort: 0.DETAIL(a)
Cross block of 2 entries  Q1
  Cross block entry 1
    Conjunct: 1.NUMTYP = 'T_TYP'
    Index only retrieval of relation 1:T1
      Index name  IDX_EMAP_1 [0:0]
  Cross block entry 2
    Leaf#01 NdxOnly 0:T2 Card=1671  Bitmapped scan
      Bool: 0.TYP = 1.PID
      FgrNdx  IDX_BSKT_0 [0:0] Fan=29
      BgrNdx1 IDX_BSKT_1 [1:1] Fan=48
        Keys: 0.TYP = 1.PID
 B.DETAIL
       11
1 row selected

! the bugcheck occurs at the exit time
exit
%RDMS−I−BUGCHKDMP, generating bugcheck dump file _[directory]RDSBUGCHK.DMP;
```

Disabling bitmapped scan (SET FLAGS 'NOBITMAPPED_SCAN) eliminates the problem.

```
set flags 'nobitmapped_scan';
select distinct DETAIL  from T2 B, T1 M
where B.TYP = M.PID and
      M.NUMTYP = 'T_TYP' ;
Tables:
  0 = T2
  1 = T1
Reduce: 0.DETAIL
Sort: 0.DETAIL(a)
```

```
Cross block of 2 entries  Q1
  Cross block entry 1
    Conjunct: TRIM (BOTH ' ' FROM 1.NUMTYP) = 'T_TYP'
    Index only retrieval of relation 1:T1
      Index name  IDX_EMAP_1 [0:0]
  Cross block entry 2
    Leaf#01 NdxOnly 0:T2 Card=1671
      Bool: 0.TYP = 1.PID
      FgrNdx  IDX_BSKT_0 [0:0] Fan=29
      BgrNdx1 IDX_BSKT_1 [1:1] Fan=48
        Keys: 0.TYP = 1.PID
 B.DETAIL
        2
        3
       11
3 rows selected
```

Dropping the foreground index also makes the query return correctly.

```
drop index IDX_BSKT_0;

set flags 'bitmapped_scan';
select distinct DETAIL  from T2 B, T1 M
where B.TYP = M.PID and
      M.NUMTYP = 'T_TYP' ;

Tables:
  0 = T2
  1 = T1
Reduce: 0.DETAIL
Sort: 0.DETAIL(a)
Cross block of 2 entries  Q1
  Cross block entry 1
    Conjunct: 1.NUMTYP = 'T_TYP'
    Index only retrieval of relation 1:T1
      Index name  IDX_EMAP_1 [0:0]
  Cross block entry 2
    Leaf#01 NdxOnly 0:T2 Card=1671  Bitmapped scan
      Bool: 0.TYP = 1.PID
      BgrNdx1 IDX_BSKT_1 [1:1] Fan=48
        Keys: 0.TYP = 1.PID
 B.DETAIL
        2
        3
       11
3 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.1.28 Query Runs Very Slow When Using Bitmapped Scan

Bug 10297647

In prior releases of Oracle Rdb, the following query runs very slow using Bitmapped Scan when one of the joined tables contains over 6 million rows.

```
set flags 'bitmapped_scan';
select 1
from T1
```

```
where
    I_ASN is not null
    and I_MCAT is not null
    and not exists (select *
                    from T2
                    where T1.I_ASN = T2.I_ASN
                        and T1.I_MCAT = T2.I_MCAT);
Tables:
  0 = T1
  1 = T2
Conjunct: <agg0> = 0
Match    (Agg Outer Join)  Q1
  Outer loop
  Match_Keys:0.I_MCAT, 0.I_ASN
    Sort: 0.I_MCAT(a), 0.I_ASN(a)
    Leaf#01 BgrOnly 0:T1 Card=6311566 Bitmapped scan
      Bool: NOT MISSING (0.I_ASN) AND NOT MISSING (0.I_MCAT)
      BgrNdx1 IX_DCAT_MCAT [0:1] Fan=39
        Keys: NOT MISSING (0.I_MCAT)
      BgrNdx2 IX_DCAT_ASN_TST_REC_TYP_A_V [0:1] Fan=42
        Keys: NOT MISSING (0.I_ASN)
  Inner loop
  Match_Keys:1.I_MCAT, 1.I_ASN
    Aggregate: 0:COUNT-ANY (<subselect>) Q2
    Sort: 1.I_MCAT(a), 1.I_ASN(a)
    Conjunct: NOT MISSING (1.I_MCAT)
    Leaf#02 BgrOnly 1:T2 Card=3325      Bitmapped scan
      Bool: NOT MISSING (1.I_ASN)
      BgrNdx1 IX_MCAT_ASN [0:1] Fan=39
        Keys: NOT MISSING (1.I_ASN)
      BgrNdx2 IX_MCAT_PK [0:1] Fan=39
        Keys: NOT MISSING (1.I_MCAT)
0 rows selected
show stat

                  process statistics at 13-DEC-2010 11:32:55.44
         elapsed time =   0 00:01:08.14              CPU time =   0 00:00:04.37
     page fault count = 1061             pages in working set = 52928
   buffered I/O count = 81                   direct I/O count = 8563
      open file count = 8                 file quota remaining = 1992
           locks held = 149                    locks remaining = 31851
      CPU utilization = 6.4%              AST quota remaining = 995
```

Disabling bitmapped scan (SET FLAGS 'NOBITMAPPED_SCAN) eliminates the problem.

```
set flags 'nobitmapped_scan';
select 1
from TB_DCAT
where
    I_ASN is not null
    and I_MCAT is not null
    and not exists (select *
                    from TB_MCAT
                    where TB_DCAT.I_ASN = TB_MCAT.I_ASN
                        and TB_DCAT.I_MCAT = TB_MCAT.I_MCAT);
Tables:
  0 = TB_DCAT
  1 = TB_MCAT
Conjunct: <agg0> = 0
Match    (Agg Outer Join)  Q1
  Outer loop
  Match_Keys:0.I_MCAT, 0.I_ASN
```

```
    Sort: 0.I_MCAT(a), 0.I_ASN(a)
    Leaf#01 BgrOnly 0:TB_DCAT Card=155000
      Bool: NOT MISSING (0.I_ASN) AND NOT MISSING (0.I_MCAT)
      BgrNdx1 IX_DCAT_MCAT [0:1] Fan=20
        Keys: NOT MISSING (0.I_MCAT)
      BgrNdx2 IX_DCAT_ASN_TST_REC_TYP_A_V [0:1] Fan=11
        Keys: NOT MISSING (0.I_ASN)
  Inner loop
  Match_Keys:1.I_MCAT, 1.I_ASN
    Aggregate: 0:COUNT-ANY (<subselect>) Q2
    Sort: 1.I_MCAT(a), 1.I_ASN(a)
    Conjunct: NOT MISSING (1.I_MCAT)
    Leaf#02 BgrOnly 1:TB_MCAT Card=3336
      Bool: NOT MISSING (1.I_ASN)
      BgrNdx1 IX_MCAT_ASN [0:1] Fan=20
        Keys: NOT MISSING (1.I_ASN)
      BgrNdx2 IX_MCAT_PK [0:1] Fan=20
        Keys: NOT MISSING (1.I_MCAT)
0 rows selected
show stat

                  process statistics at 13-DEC-2010 11:31:47.29
         elapsed time =   0 00:01:48.09              CPU time =   0 00:00:01.72
     page fault count = 49               pages in working set = 35584
   buffered I/O count = 114                  direct I/O count = 8655
      open file count = 8                 file quota remaining = 1992
           locks held = 149                    locks remaining = 31851
      CPU utilization = 1.5%               AST quota remaining = 995
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.1.29 Query With "NOT (conj1 OR conj2 OR conj3)" Predicate Bugchecks

Bug 11850015

In prior releases of Oracle Rdb, the following query with "NOT (conj1 OR conj2 OR conj3)" bugchecks at setup_for_no_get.

```
select * from all_cat where
    t_type <> 'SEQ' and t_type <> 'SYN' and
    not (owner = 'MSYS' OR owner = 'OSYS' OR owner = 'WSYS');
%RDMS-I-BUGCHKDMP, generating bugcheck dump file _[directory]RDSBUGCHK.DMP;
```

This query worked in Oracle Rdb Release 7.2.4.1 and started breaking in Oracle Rdb Release 7.2.4.2 due to a fix for Bug 9509316.

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.1.30 Query Returns Wrong Results Using Bitmap Scan With Zigzag Match

Bug 8344512

When bitmapped scan is used on sorted ranked indices, this metadata query returns no results.

```
Tables:
  0 = RDB$CONSTRAINT_RELATIONS
  1 = RDB$CONSTRAINTS
  2 = RDB$RELATION_CONSTRAINTS
Reduce: 0.RDB$CONSTRAINT_NAME
Cross block of 2 entries  Q1
  Cross block entry 1
    Conjunct: 1.RDB$CONSTRAINT_NAME = 0.RDB$CONSTRAINT_NAME
    Match  Q1
      Outer loop      (zig-zag)
      Match_Key:0.RDB$CONSTRAINT_NAME
      Index_Key:RDB$CONSTRAINT_NAME
        Leaf#01 Sorted 0:RDB$CONSTRAINT_RELATIONS Card=8         Bitmapped scan
          Bool: 0.RDB$RELATION_NAME = <var0>
          FgrNdx  RDB$CR_CONSTRAINT_NAME_NDX [0:0] Fan=8
          BgrNdx1 RDB$CR_REL_NAME_NDX [1:1] Fan=8
            Keys: 0.RDB$RELATION_NAME = <var0>
      Inner loop      (zig-zag)
      Match_Key:1.RDB$CONSTRAINT_NAME
      Index_Key:RDB$CONSTRAINT_NAME
        Get     Retrieval by index of relation 1:RDB$CONSTRAINTS
          Index name  RDB$CON_CONSTRAINT_NAME_X [0:0]
  Cross block entry 2
    Conjunct: <agg0> = 0
    Aggregate-F1: 0:COUNT-ANY (<subselect>) Q2
    Index only retrieval of relation 2:RDB$RELATION_CONSTRAINTS
      Index name  RDB$RLC_CONSTRAINT_NAME_NDX [1:1]  Direct lookup
        Keys: 1.RDB$CONSTRAINT_NAME = 2.RDB$CONSTRAINT_NAME


This script works with SORTED index:

set verify;
drop database
    filename pers;
import database
    from 'test$db_source:personnel_sql'
    filename 'pers'
    system index (type is sorted);
set flags 'strategy,detail(2)';
show table (constraint) employees
disconnect all;

This script fails with RANKED index:

set verify;
drop database
    filename pers;
import database
    from 'test$db_source:personnel_sql'
    filename 'pers'
    system index (type is sorted ranked);
set flags 'bitmapped_scan,strategy,detail(2)';
show table (constraint) employees
disconnect all;
```

The problem can also be reproduced using either RDO or SQL, as can be seen in the following scripts:

```
FOR
```

```
    CR IN Rdb$CONSTRAINT_RELATIONS
    CROSS C IN Rdb$CONSTRAINTS
    OVER Rdb$CONSTRAINT_NAME
    WITH CR.Rdb$RELATION_NAME = 'EMPLOYEES' AND
        NOT ANY RC IN RdbVMS$RELATION_CONSTRAINTS
        WITH C.RDB$CONSTRAINT_NAME = RC.RDB$CONSTRAINT_NAME
print C.Rdb$CONSTRAINT_NAME
       ,C.Rdb$CONSTRAINT_SOURCE
END_FOR

SQL script:

select C.Rdb$CONSTRAINT_NAME,C.Rdb$CONSTRAINT_SOURCE
  FROM Rdb$CONSTRAINT_RELATIONS CR
       JOIN
       Rdb$CONSTRAINTS C
       ON CR.Rdb$CONSTRAINT_NAME = C.Rdb$CONSTRAINT_NAME
  where
      CR.Rdb$RELATION_NAME = 'EMPLOYEES' AND
      NOT EXISTS
      (select * from RdbVMS$RELATION_CONSTRAINTS RC
        where C.RDB$CONSTRAINT_NAME = RC.RDB$CONSTRAINT_NAME);
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.1.31 Query With Over 26 Million Rows Slows Down

Bug 9454843

In prior releases of Oracle Rdb, a query became very slow due to a million blocks being allocated for a sort that is done repeatedly that usually involves only a handful of records.

One of the tables has over 26 million rows, but there are nearly 6 million values for DEBT_ID so the average number of rows to sort is under 5.

The following is the simplified version of the query.

```
select *
from
    DEBT as C2,
    DEBT as C33
    left outer join
    (select C36.DEBT_ID, C36.TRANSACTION_ID, sum(
        C36.CURRENT_BALANCE_AMOUNT)
        from BALANCE_HISTORY C36
        group by C36.DEBT_ID, C36.TRANSACTION_ID)
    as C34 (F1, F2, F3)
         on (C34.F1 = C33.DEBT_ID)
            and (C34.F2 = (select C52.TRANSACTION_ID from
                    BALANCE_HISTORY C52
                where (((C52.DEBT_ID = C33.DEBT_ID)
                    and (C52.BALANCE_TYPE_CODE = 'DBC'))
                        )
                order by C52.CHANGE_TIME desc
                limit to 1 rows))
    where
        (C33.DEBT_ID = C2.DEBT_ID);
Tables:
```

```
  0 = DEBT
  1 = DEBT
  2 = BALANCE_HISTORY
  3 = BALANCE_HISTORY
Conjunct: 1.DEBT_ID = 0.DEBT_ID
Match  Inner_TTBL Q1
  Outer loop      (zig-zag)
  Match_Key:0.DEBT_ID
  Index_Key:DEBT_ID
    Get     Retrieval by index of relation 0:DEBT
      Index name  DEBT_PK [0:0]
  Inner loop
  Match_Key:1.DEBT_ID
    Temporary relation
    Cross block of 2 entries        (Left Outer Join)  Q4
      Cross block entry 1
        Cross block of 2 entries  Q6
          Cross block entry 1
            Get     Retrieval by index of relation 1:DEBT
              Index name  DEBT_PK [0:0]
          Cross block entry 2
            Aggregate: 0:VIA (3.TRANSACTION_ID) Q5
            Firstn: 1
            Sort: 3.CHANGE_TIME(d)
              SortId# 4., # Keys 2
                Item# 1, Dtype: 2, Order: 1, Off: 0, Len: 1
                Item# 2, Dtype: 35, Order: 1, Off: 1, Len: 8
                LRL: 40, NoDups:0, Blks:64423, EqlKey:0, WkFls: 2
            Leaf#01 BgrOnly 3:BALANCE_HISTORY Card=26384512
              Bool: (3.DEBT_ID = 1.DEBT_ID) AND (3.BALANCE_TYPE_CODE = 'DBC')
              BgrNdx1 BALANCE_HISTORY_PK [2:2] Fan=9
                Keys: (3.DEBT_ID = 1.DEBT_ID) AND (3.BALANCE_TYPE_CODE = 'DBC')
      Cross block entry 2
        Conjunct: 2.DEBT_ID = 1.DEBT_ID
        Merge of 1 entries  Q4
          Merge block entry 1  Q2
          Aggregate: 1:SUM (2.CURRENT_BALANCE_AMOUNT) Q3
          Sort: 2.DEBT_ID(a), 2.TRANSACTION_ID(a)
            SortId# 5., # Keys 4
              Item# 1, Dtype: 2, Order: 0, Off: 0, Len: 1
              Item# 2, Dtype: 8, Order: 0, Off: 1, Len: 4
              Item# 3, Dtype: 2, Order: 0, Off: 5, Len: 1
              Item# 4, Dtype: 8, Order: 0, Off: 6, Len: 4
              LRL: 32, NoDups:0, Blks:1000000, EqlKey:0, WkFls: 2
          Leaf#02 BgrOnly 2:BALANCE_HISTORY Card=26384512
            BgrNdx1 BALANCE_HISTORY_PK [1:1] Fan=9
              Keys: 2.DEBT_ID = 1.DEBT_ID
0 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.2 SQL Errors Fixed

3.2.1 Unexpected Bugcheck When Using INSERT ... SELECT Into a View

Bug 9383578

In prior releases of Oracle Rdb, it was possible that using INSERT ... SELECT into a view based on another view would generate a bugcheck dump. This occurred when the base table contained DEFAULT values for some columns.

The following example shows the error reported by a customer.

```
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]RDSBUGCHK.DMP;
%RDB-E-INVALID_BLR, request BLR is incorrect at offset 1041
-RDMS-E-UNKNOWN_VAR, unknown variable 2 found in the query string
```

This problem could also occur when an INSERT was contained in a FOR cursor loop in a compound statement.

This problem has been corrected in Oracle Rdb Release 7.2.5. Rdb now correctly determines the context for the nested table when processing the DEFAULT value or AUTOMATIC INSERT AS values for a base table referenced by nested views.

3.2.2 Warning Now Issued for Unsupported Character Operations

This release of Oracle Rdb changes the behavior of the LIKE ... IGNORE CASE, UPPER and LOWER functions in the cases where they are applied to character string expressions with a character set that does not support casing (upper and lower case equivalent characters).

In previous versions of Oracle Rdb, UPPER and LOWER were accepted for these character sets but they are now ignored by SQL. That is, the UPPER and LOWER functions are discarded if the character set does not have upper and lower case characters. This would be applicable to a character set such as KANJI.

In previous versions LIKE ... IGNORE CASE would cause the query to fail if the character string expressions did not support casing. SQL now ignores this clause with an issued warning.

The following example shows the issued warnings.

```
SQL> create table kan ( a char(10) character set kanji);
SQL> select * from kan where a like _kanji'aa' ignore case;
%SQL-W-NOCASING, IGNORE CASE not supported for character set KANJI - ignored
0 rows selected
SQL> select * from kan where upper(a) = _kanji'aa';
%SQL-W-NOCASING, UPPER not supported for character set KANJI - ignored
0 rows selected
SQL> select * from kan where lower(a) = _kanji'aa';
```

```
%SQL-W-NOCASING, LOWER not supported for character set KANJI - ignored
0 rows selected
SQL> rollback;
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.2.3 Incorrect Results From LIKE ... IGNORE CASE

In prior releases of Oracle Rdb, the LIKE ... IGNORE CASE predicate could return the wrong results if the pattern string included the "*" character.

The following example shows the incorrect results from LIKE ... IGNORE CASE. The LIKE ... IGNORE CASE example should have returned only two rows.

```
SQL> create table like_test
cont>     (TEXT varchar(10)
cont>     );
SQL> insert into like_test (TEXT) values ('abc');
1 row inserted
SQL> insert into like_test (TEXT) values ('abc*');
1 row inserted
SQL> insert into like_test (TEXT) values ('ABC');
1 row inserted
SQL> insert into like_test (TEXT) values ('ABC*');
1 row inserted
SQL>
SQL>
SQL> select * from like_test;
 TEXT
 abc
 abc*
 ABC
 ABC*
4 rows selected
SQL>
SQL> -- An ordinary, case sensitive, LIKE gets the proper results:
SQL>
SQL> select * from like_test where text like 'a%*%';
 TEXT
 abc*
1 row selected
SQL>
SQL> -- But when you IGNORE CASE things get weird:
SQL>
SQL> select * from like_test where text like 'a%*%' ignore case;
 TEXT
 abc
 abc*
 ABC
 ABC*
4 rows selected
SQL>
SQL> rollback;
```

Additionally, LIKE ... IGNORE CASE was ignoring diacritial markings, contrary to the LIKE documentation.

At the same time, several restrictions previously in place for this predicate have been removed.

```
SQL> select * from like_test where text like pattern || '%' ignore case;
%SQL-F-BADCOLIGNCAS, The LIKE pattern is incompatible with IGNORE CASE
SQL> select * from like_test where text like pattern ignore case;
%SQL-F-BADCOLIGNCAS, The LIKE pattern is incompatible with IGNORE CASE
SQL> declare :pat varchar(10) = 'a%*%';
SQL> begin
cont> for :l as select *
cont>     from like_test
cont>     where text like :pat ignore case
cont> do
cont>     trace :l.text;
cont> end for;
cont> end;
%SQL-F-MSP_LIKE_XLATE, Translation of LIKE pattern string not supported in a
compound statement
```

The predicate pattern can now be any expression (including a column reference) and may be used within a compound statement.

Oracle Rdb now supports a more general LIKE ... IGNORE CASE implementation which corrects these problems and lifts these restrictions.

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.2.4 Unexpected ACCVIO When Using Dynamic DECLARE Cursor Statement

Bug 9166313

In prior releases of Oracle Rdb, if a database or a user had transaction modes restricted to READ ONLY, it was possible for the SQL dynamic DECLARE of a list cursor to fail with an ACCVIO.

```
Declare list cursor SQLCODE = -1
 ** The parameter value returned is: 8347456
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=0000000000000001C, PC=00000000005851AC, PS=0000001B
```

A database can be altered to allow only READ ONLY transactions, as shown in this example:

```
alter database
    filename PERSONNEL
    set transaction modes (read only);
```

All users of such a database could see this problem when using dynamically declared cursors such as those used by ODBC, JDBC, SQL/Services or OCI Services for Rdb.

A single user can be assigned a profile that restricts the transactions to READ ONLY, as shown in this example:

```
create profile RO_PROFILE
    default transaction READ ONLY
    transaction modes (READ ONLY);

create user SAMPLE_USER
    identified externally
```

```
    profile RO_PROFILE;
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.2.5 Incorrect Value Returned By RETURNING Clause of the INSERT Statement

Bug 10008867

In prior releases of Oracle Rdb, it was possible that using the RETURNING clause of the INSERT statement would evaluate an AUTOMATIC AS expression a second time, once for the INSERT and again if they were returned by the RETURNING clause. This could be problematic if the expression called a function that had side effects, such as updating the SEQUENCE, or performing some external action.

The following example shows that the value returned by the RETURNING clause is different from what was actually inserted.

```
SQL> declare :v1 integer;
SQL> declare :v1_ind integer;
SQL>
SQL> create sequence s1;
SQL>
SQL> create module m1
cont>     language SQL
cont>
cont>     function f1()
cont>       returns integer
cont>       not deterministic;
cont>     return s1.nextval;
cont>
cont> end module;
SQL>
SQL> create table t1
cont>     (c1 automatic insert as f1()
cont>     ,c2 char(10));
SQL>
SQL>insert into t1 (c2) values ('test1') returning c1 into :v1 indicator
:v1_ind;
1 row inserted
SQL> print :v1 indicator :v1_ind;
          V1
           2
SQL> select c1 from t1;
          C1
           1
1 row selected
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.2.6 Unexpected Failure When Adding IDENTITY Columns

In previous releases of Oracle Rdb, using a delimited table name could cause the CREATE or ALTER TABLE statement to fail when an IDENTITY column was defined.

The following example shows this problem.

```
SQL> CREATE TABLE "Emps" (
cont>     "Id" int IDENTITY PRIMARY KEY ,
cont>     "Name" NCHAR VARYING(2000)
cont> );
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-E-SEQNEXTS, sequence "Emps" does not exist in this database
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.5. Oracle Rdb was incorrectly uppercasing the name of the identity sequence.

3.2.7 Unexpected Bugcheck Dump Produced When UNION and GROUP BY Are Used

Bug 9952060

In prior releases of Oracle Rdb, it was possible in rare cases to have a bugcheck dump produced when a UNION also included a GROUP BY. The ON clause of a JOIN needs to reference the same GROUP BY expression in a subsequent UNION branch.

The following example shows the problem. The ON clause in the second part of the UNION uses the same GROUP BY expression as the first part of the UNION.

```
SQL> select substring (jh.employee_id from 1 for 5)
cont>  FROM job_history jh
cont>  JOIN salary_history sh
cont>    ON sh.employee_id = substring (jh.employee_id from 1 for 5)
cont>  GROUP BY substring (jh.employee_id from 1 for 5)
cont> UNION ALL
cont> SELECT substring (jh.employee_id from 1 for 5)
cont>  FROM job_history jh
cont>  JOIN salary_history sh
cont>    ON sh.employee_id = substring (jh.employee_id from 1 for 5)
cont> GROUP BY substring (jh.employee_id from 1 for 5)
cont> ;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=00000000000000D4, PC=FFFFFFFF8027E8C1, PS=0000001B
```

This problem has been corrected in Oracle Rdb Release 7.2.5. SQL now correctly processes the ON clause in these cases.

3.2.8 SET EXECUTE Now Implicitly Executed When ROLLBACK Question Is Asked

In prior versions of Oracle Rdb, Interactive SQL would ask if you wished to ROLLBACK any changes made to the database prior to exiting. However, if the command SET NO EXECUTE had previously been executed, any such ROLLBACK was ignored. With this release of Rdb, a SET EXECUTE is implicitly executed before returning control back to Interactive SQL. Therefore, any commands executed to terminate the session will be executed.

The following example shows this.

```
SQL> set no execute;
SQL>
SQL> select * from tt;
0 rows selected
SQL> exit
There are uncommitted changes to this database.
Would you like a chance to ROLLBACK these changes (No)? y
SQL> select * from tt;
           A
        NULL
1 row selected
SQL> rollback;
```

3.2.9 Unexpected Bugcheck When Accessing View Changed Using the ALTER VIEW Statement

In prior releases of Oracle Rdb, the ALTER VIEW command may set the column RDB$DBKEY_LENGTH in the system table RDB$RELATIONS to the wrong value. This may lead to a bugcheck with a footprint similar to the one shown here.

- SYSTEM−F−ACCVIO, access violation
- Exception occurred at RDMSHRP721\RDMS$$PRE_EXECUTION + 000002F1
- Called from RDMSHRP721\RDMS$$COMPILE_FOR_IF + 00004340
- Called from RDMSHRP721\RDMS$$COMPILE_STMT + 000014C0
- Called from RDMSHRP721\RDMS$$COMPILE_STMT + 00000440

The VIEW definition can be removed using the DROP VIEW statement and then recreated using the CREATE VIEW statement, or you can repeat the ALTER VIEW statement after this release has been installed.

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.2.10 Unexpected CAPTIVEACCT Error When Using Spawn Directive in Interactive SQL for RESTRICTED Accounts

Bug 10353927

In prior releases of Oracle Rdb, an account marked as RESTRICTED was not permitted to execute the SPAWN directive in Interactive SQL. The result is shown in the following example:

```
SQL> $ show time
%COSI−F−CAPTIVEACCT, captive account − can't create sub−process
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.5. Interactive SQL no longer treats the OpenVMS UAF flags CAPTIVE and RESTRICTED as identical. As described by the OpenVMS documentation, a RESTRICTED account may still access DCL. This change brings interactive SQL into conformance with other OpenVMS utilities such as MAIL or EDIT/TPU which allow SPAWN actions.

3.2.11 Unexpected NOTRIGRTN Error When Trigger Calls a Procedure Using LOCK TABLE Statement

Bug 11789653

In prior releases of Oracle Rdb, attempts to create a trigger that called a stored procedure which used the LOCK TABLE statement would fail with the error NOTRIGRTN "this stored routine may not be called from a trigger". The following example shows this problem.

```
SQL> create module MY_MODULE
cont>     language      SQL
cont>
cont>     procedure MY_PROC
cont>           (in :IN_ID CHAR(5)
cont>           );
cont>     begin
cont>     declare :EMP_ID CHAR(5);
cont>
cont>     lock table SALARY_HISTORY for PROTECTED WRITE mode;
cont>
cont>     for :EMP_REC
cont>          as each row of table cursor EMP_CURSOR for
cont>             select EMPLOYEE_ID, LAST_NAME, FIRST_NAME
cont>             from EMPLOYEES
cont>             where EMPLOYEE_ID = :IN_ID
cont>     do
cont>         set :EMP_ID = :EMP_REC.EMPLOYEE_ID;
cont>         insert into SALARY_HISTORY
cont>            values (:EMP_ID, 100.00, current_timestamp, null);
cont>     end for;
cont>     end;
cont> end module;
SQL>
SQL> create trigger DEGREE_UPDATE
cont>     after UPDATE of DEGREE on DEGREES
cont>     referencing OLD as ODEG NEW as NDEG
cont>     (call MY_PROC (NDEG.EMPLOYEE_ID))
cont>     for each row;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-RTN_FAIL, routine "MY_PROC" failed to compile or execute successfully
-RDMS-E-NOTRIGRTN, this stored routine may not be called from a trigger
SQL>
```

The only statements which should cause this error are: SET TRANSACTION, START TRANSACTION, COMMIT, COMMIT AND CHAIN, ROLLBACK and ROLLBACK AND CHAIN.

This problem has been corrected in Oracle Rdb Release 7.2.5. However, the procedure which references the LOCK TABLE statement will need to be recreated before it can be referenced by the trigger definition.

3.2.12 Unexpected Bugchecks When Some Undocumented Syntax Used

Bugs 3949015, 7413895, and 7655283

In prior releases of Oracle Rdb, the incomplete and undocumented SIMILAR TO operator generated a bugcheck when used. This prototype code was erroneously included in the production release.

This problem has been corrected in Oracle Rdb Release 7.2.5. References will now generate a WISH_LIST error.

```
SQL> select * from employees where last_name similar to 'S[a-z]*';
%RDB-F-WISH_LIST, feature has not been implemented
SQL>
```

3.2.13 Unexpected Slow Performance for Query Using SQL Functions

Bug 9112403

In prior releases of Oracle Rdb, a query might exhibit slow performance on the first execution in a session. This occurred when a SQL function was called that contained only constant literal arguments, for instance, TO_DATE('20080625194943','YYYYMMDDHH24MISS'). In such cases, the function has not been compiled and therefore cannot be used to resolve column value for the optimizer. Rdb would then assume that the function was NOT DETERMINISTIC and thus execute the function for every row in the table. Subsequent executions of the same or similar query would perform better because the function, now compiled, could be removed from the main loop and executed just once.

This problem has been corrected in Oracle Rdb Release 7.2.5. The Oracle Rdb query compiler now pre-processes queries to extract and compile any referenced SQL functions. This allows their execution during the compile phase for the query.

A workaround to this problem for older releases is to perform a simple query to load and compile the function prior to execution of the main query, as in the following example.

```
EXEC SQL
    begin
    declare :dt date;
    set :dt = TO_DATE('20080625194943','YYYYMMDDHH24MISS');
    end;
```

3.3 RDO and RDML Errors Fixed

3.3.1 Duplicate Values Generated For IDENTITY Column When RDO Interface Used For STORE

Bug 11804365

In prior releases of Oracle Rdb, it was possible that an RDBPRE (RDO) application would insert duplicate values for an IDENTITY column.

The following example shows that a STORE statement nested in a FOR loop was treated as a single statement when generating the IDENTITY sequence's next value (NEXTVAL). Naturally, neither RDO nor RdbPRE Pre−compiler establish SQL semantics.

```
RDO> for o in OUTER_T
cont>     store i in INNER_T using
cont>         i.tag = 1
cont>     end_store
cont>     store i in INNER_T using
cont>         i.tag = 2
cont>     end_store
cont> end_for
RDO>
RDO> for i in INNER_T
cont>     print i.ident_column, i.tag
cont> end_for
 IDENT_COLUMN           TAG
            1             1
            1             2
RDO>
```

However, the definition of an IDENTITY column is independent of the interface used and should be evaluated for each STORE statement.

This problem has been corrected in Oracle Rdb Release 7.2.5. Oracle Rdb now detects that an IDENTITY sequence is used by a STORE statement and updates it correctly.

Note

Semantics described for the SQL language related to sequences are not present in the RDO language. Therefore, an RDO or RdbPRE Pre−compiler application may not behave in the same way as a similar SQL, SQL Module Language, or SQL Pre−compiler application. For instance, a table that has an AUTOMATIC INSERT AS column that evaluates a sequence's NEXTVAL will not be updated within a similar FOR loop to the example shown above.

3.4 RMU Errors Fixed

3.4.1 RMU/UNLOAD to XML Does Not Replace Special Characters

Bug 9597122

When using RMU/UNLOAD to create an XML file, certain special characters were being inserted as the actual character, not as an XML special character sequence. These special characters include & < > ' and ".

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.4.2 RMU/RESTORE Could Fail When /BLOCKS_PER_PAGE Was Specified

Bug 8869988

The /BLOCKS_PER_PAGE qualifier can be specified with the Oracle Rdb RMU/RESTORE command to specify a new page size for a database storage area. There was a problem where the restore of a uniform storage area could fail when a new /BLOCKS_PER_PAGE value was specified. The logical area page clump lists on the uniform area spam pages could be corrupted when a new BLOCKS_PER_PAGE value was specified. Pages could be assigned to the wrong logical area in a page clump list if RMU/RESTORE tried to change the clumps per page value on the spam pages in the uniform area based on the new page size. For the RMU/RESTORE of uniform areas, the spam page clumps per page value cannot be changed during an RMU/RESTORE.

This problem resulted in error messages during the restore due to pages being assigned to the wrong logical areas as well as fatal errors and bugcheck dumps due to the invalid spam pages. An RMU/VERIFY of the uniform storage areas would also report logical area errors due to the spam page corruption.

This problem has been fixed. Now if a new BLOCKS_PER_PAGE value is specified for the RMU/RESTORE of a uniform storage area, RMU/RESTORE will never change the clumps per page value for the SPAM pages in that area. Also, new checks have been added to RMU/RESTORE to return an error and terminate an RMU/RESTORE at the start of the restore if a new BLOCKS_PER_PAGE value is specified for a uniform area that would cause database corruption. The user can then repeat the RMU/RESTORE either without specifying a new BLOCKS_PER_PAGE value or by specifying a BLOCKS_PER_PAGE value which RMU/RESTORE will accept since it will not cause database corruption of the uniform storage area.

The following example shows the problem. RMU/RESTORE would put out logical area diagnostics for the logical areas in uniform storage areas for which a new BLOCKS_PER_PAGE value was specified and a fatal error would usually result which could vary but would be related to the corrupted spam pages. Usually a fatal error would result in a bugcheck dump file being created. Note that this problem only happened if RMU/RESTORE decided to change the spam page clumps per page value based on the new blocks per page value, otherwise the restore succeeded and did not put out logical area errors or corrupt the spam pages.

```
$ RMU/RESTORE/LOG/NOCDD/NORECOVER/DIR=SYS$DISK:[] -
  /OPTIONS=SYS$INPUT TEST_DATABASE.RBF
```

```
AREA1/BLOCKS_PER_PAGE=16
AREA2/BLOCKS_PER_PAGE=16
%RMU-I-AIJRSTBEG, restoring after-image journal "state" information
%RMU-I-AIJRSTEND, after-image journal "state" restoration complete
%RMU-I-RESTXT_00, Restored root file
  DISK:[DIRECTORY]TEST_DATABSE.RDB;1
%RMU-I-RESTXT_18, Processing options file SYS$INPUT
area1/blocks_per_page=16
 area1/blocks_per_page=16
area2/blocks_per_page=16
 area2/blocks_per_page=16
%RMU-I-RESTXT_21, Starting full restore of storage area
 (RDB$SYSTEM) DISK:[DIRECTORY]SYS.RDA;1
  at 24-JUN-2010 15:48:24.05
%RMU-I-RESTXT_21, Starting full restore of storage area
 (AREA1) DISK:[DIRECTORY]AREA1.RDA;1
  at 24-JUN-2010 15:48:24.18
%RMU-W-BADPTLARE, invalid larea for uniform data page 5 in storage
area 8
%RMU-W-BADPTLAR2,                   SPAM larea_dbid: 97, page
larea_dbid: 117
%RMU-W-BADPTLARE, invalid larea for uniform data page 11 in storage
area 8
%RMU-W-BADPTLAR2,                   SPAM larea_dbid: 97, page
larea_dbid: 117
%RMU-W-BADPTLARE, invalid larea for uniform data page 719 in storage
area 8
%RMU-W-BADPTLAR2,                   SPAM larea_dbid: 117, page
larea_dbid: 97
%RMU-I-RESTXT_24, Completed full restore of storage area (RDB$SYSTEM)
DISK:[DIRECTORY]SYS.RDA;1 at 24-JUN-2010 15:48:26.49
%RMU-I-RESTXT_24, Completed full restore of storage area (AREA1)
DISK:[DIRECTORY]PMR_DATA.RDA;1 at 24-JUN-2010 15:48:40.03
%SYSTEM-F-ROPRAND, reserved operand fault at PC=00000000802C1642,
PS=0000001B
%RMU-I-BUGCHKDMP, generating bugcheck dump file
DISK:[DIRECTORY]RMUBUGCHK.DMP;
%RMU-F-FATALERR, fatal error on RESTORE
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 24-JUN-2010
15:48:43.18
```

The following example shows the new error messages which can now be returned if a BLOCKS_PER_PAGE value is specified with the RMU/RESTORE command which will cause database corruption for a uniform storage area.

```
$ RMU/RESTORE/LOG/NOCDD/NORECOVER/DIR=SYS$DISK:[]-
 DISK:[DIRECTORY]TEST_DATABASE.RBF -
AREA1/BLOCKS_PER_PAGE=63, AREA2/BLOCKS_PER_PAGE=63
RMU-I-AIJRSTBEG, restoring after-image journal "state" information
%RMU-I-AIJRSTEND, after-image journal "state" restoration complete
%RMU-I-RESTXT_00, Restored root file
 DISK:[DIRECTORY]TEST_DATABASE.RDB;1
%RMU-W-CLMPPGCNT, the clump page count multiplied by the number of
  blocks
%RMU-W-CLMPPGCN2,                   per page is greater than the
  maximum of 64 blocks
  %RMU-W-CLMPPGCN3,                   Computed: 189. CLUMP_PAGCNT = 3;
  PAG_BLKCNT = 63
%RMU-F-BDCLMPPGCNT, The specified BLOCKS_PER_PAGE value would cause
  an illegal clump page count for storage area
  DISK:[DIRECTORY]AREA1.RDA;1
```

```
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at  2-JUL-2010
 14:07:03.74
$ RMU/RESTORE/LOG/NOCDD/NORECOVER/DIR=SYS$DISK:[] -
  /OPTIONS=SYS$INPUT DISK:[DIRECTORY]MFP.RBF
MF_PERS_SEGSTR/BLOCKS_PER_PAGE=8
%RMU-I-AIJRSTBEG, restoring after-image journal "state" information
%RMU-I-AIJRSTEND, after-image journal "state" restoration complete
%RMU-I-RESTXT_00, Restored root file
 DISK:[DIRECTORY]MF_PERSONNEL.RDB;1
  %RMU-I-RESTXT_18, Processing options file SYS$INPUT
MF_PERS_SEGSTR/blocks_per_page=8
 MF_PERS_SEGSTR/blocks_per_page=8
%RMU-F-BUFSMLPAG, The specified BLOCKS_PER_PAGE 8 exceeds the
 buffer size 6 for storage area
 DISK:[DIRECTORY]MF_PERS_SEGSTR.RDA;1
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at  2-JUL-2010
 14:36:52.26
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.4.3 An Incremental Instead Of a Full Backup Could Corrupt a Database

Bug 10021340

As documented in the Oracle Rdb Guide To Database Maintenance (see sections 7.5.2 and 7.5.3), a full database backup, not an incremental backup, should be done after changing the physical and logical design of an Oracle Rdb database. This includes major changes made to the structures of the database with the SQL ALTER DATABASE statement. If an incremental backup is done instead of a full backup in such cases, database corruption can occur when the incremental restore is later executed on the database.

Instead of just recommending in our documentation doing a full database backup in these cases to prevent database corruption, we now will return an error and not execute the incremental backup if an incremental backup instead of a full database backup is executed following certain major changes to the database that could result in database corruption when the incremental restore is later executed. This will protect the integrity of the database.

The following message will be output in such cases when the incremental backup is attempted.

```
$ RMU/BACKUP/INCREMENTAL/NOLOG MF_PERSONNEL MFP.RBF
%RMU-F-NOFULLBCK, no full backup of this database exists
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 19-AUG-2010
 17:17:07.03
```

The user should then do a full backup of the database. Note that the incremental backup will not be allowed only in certain cases where major changes to the database have been made which could cause database corruption later when the incremental restore is applied to the database. Once a full backup has been done in such cases, subsequent incremental backups will succeed.

The following example shows one specific instance of the problem. In this particular case, a storage area N3 was added to the database FOO after the full database backup and then an incremental backup was done instead of a full backup. This caused database corruption when the incremental restore was later applied to the FOO database.

```
$ set def DIR1:[A1]
$ SQL
create data file foo
reserve 10 storage area
create storage area n1
create storage area n2
create storage area n4;
create table t11 (i1 int);
create storage map m11 for t11 store in n4;
insert into t11 values (123);
 1 row inserted
commit;
disconnect all;
alter data file foo drop storage area n2;
exit;
$ rmu/backup/nolog foo.rdb f1.rbf
$ SQL
alter data file foo
add storage area n3;
att 'fi foo';
create table t1 (i1 int);
create storage map m1 for t1 store in n3;
insert into t1 values (123);
1 row inserted
commit;
exit;
$ rmu/backup/incremental/nolog foo.rdb f2.rbf
$ set def DIR2:[A2]
$ rmu/restore/nocdd/noafter/norec/nolog/dir=dir2:[a2] -
  dir1:[a1]f1.rbf
$ rmu/restore/nocdd/log/norec/incremental/noconf/dir=dir2:[a2] -
  dir1:[a1]f2.rbf
%RMU-I-RESTXT_00, Restored root file DIR2:[A2]FOO.RDB;1
%RMU-I-RESTXT_22, Starting incremental restore of storage area
 (RDB$SYSTEM) DIR2:[A2]FOO.RDA;1 at 12-AUG-2010 11:05:10.16
%RMU-I-RESTXT_22, Starting incremental restore of storage area (N4)
 DIR2:[A2]N4.RDA;1 at 12-AUG-2010 11:05:10.16
%RMU-I-RESTXT_22, Starting incremental restore of storage area (N3)
 DIR2:[A2]N3.RDA;1 at 12-AUG-2010 11:05:10.17
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=FFFFFFFFEB6BA336, PC=00000000003A0434, PS=0000001B
%RMU-I-BUGCHKDMP, generating bugcheck dump file
 SERDB_USER1:[HOCHULI]RMUBUGCHK.DMP;
%RMU-F-FATALERR, fatal error on RESTORE
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 12-AUG-2010
 11:05:10.28
$ rmu/verify/all foo.rdb
%RMU-F-INV_ROOT, database file has illegal format
%RMU-F-FTL_VER, Fatal error for VERIFY operation at 12-AUG-2010
 11:05:10.32
```

The following example shows the new behavior using the above example. When the incremental backup is attempted after the storage area is added, the incremental backup is not executed and the RMU-F-NOFULLBCK error message is output because a full backup is required in this case. When the user then does a full backup it succeeds. The user later does a full restore instead of a full restore followed by an incremental restore and no corruption of the database occurs.

```
$ set def DIR1:[A1]
$ SQL
create data file foo
reserve 10 storage area
```

```
create storage area n1
create storage area n2
create storage area n4;
create table t11 (i1 int);
create storage map m11 for t11 store in n4;
insert into t11 values (123);
 1 row inserted
commit;
disconnect all;
alter data file foo drop storage area n2;
exit;
$ rmu/backup/nolog foo.rdb f1.rbf
$ SQL
alter data file foo
add storage area n3;
att 'fi foo';
create table t1 (i1 int);
create storage map m1 for t1 store in n3;
insert into t1 values (123);
1 row inserted
commit;
exit;
$ rmu/backup/incremental/nolog foo.rdb f2.rbf
%RMU-F-NOFULLBCK, no full backup of this database exists
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 12-AUG-2010
 11:04:07.03
$ rmu/backup/nolog foo.rdb f2.rbf
$ set def DIR2:[A2]
$ rmu/restore/nocdd/nolog/norec/noconf/dir=dir2:[a2] -
  dir1:[a1]f2.rbf
$ rmu/verify/all foo.rdb
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.4.4 RMU/BACKUP/AFTER Invalid Open Record With Emergency AIJ Files

Bug 8943703

If RMU/BACKUP/AFTER was being executed to back up fixed AIJ files at a time when there was an extremely heavy load on an Oracle Rdb database which caused the existing AIJ files to fill up so that additional emergency fixed AIJ files had to be created and frequent switching between AIJ files was taking place, an invalid OPEN record with an invalid sequnce number of "−1" and other incorrect fields could be created in the backup AIJ file for one of the non−emergency AIJ files that was backed up. If the backed up AIJ file was used later for an RMU/RECOVER of the database, the invalid sequence number would cause the recovery to fail. Note that the invalid OPEN record was created in the backed up AIJ file, not in the live fixed AIJ files.

This problem is fixed in Oracle Rdb Release 7.2.4.0 and all later 7.2.4 releases but was not documented in previous 7.2.4 release notes. It happened because a flag in the database root for each AIJ file which indicates that the AIJ file is currently being backed up, was being cleared too soon. This flag is checked by the AIJ server code when switching between files and for other operations. If the fixed AIJ files were being heavily used during the backup, this set up a small time window when the sequence number of the OPEN record of the AIJ file currently being backed up could be reset to −1, to indicate that the AIJ file was not being currently used, just before RMU/BACKUP/AFTER read the AIJ file OPEN record and copied it to the backup AIJ file.

The following shows an example of the invalid open record when the AIJ backup file created by the RMU/BACKUP/AFTER command was dumped.

```
$ RMU/BACKUP/AFTER_JOURNAL/NOLOG/NOQUIET mf_personnel.rdb mfpaipbck
$ RMU/DUMP/AFTER/NODATA/OUT=DUMP.LIS mfpaipbck
$ TYPE DUMP.LIS

60494/131589     TYPE=O, LENGTH=510, TAD=19−JUL−2010 14:35:51.53,
CSM=00

Database DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
Database timestamp is 19−JUL−2010 11:02:31.47
Facility is "RDMSAIJ ", Version is 721.0
Database version is 72.1
AIJ Sequence Number is −1
Last Commit TSN is 0:0
Synchronization TSN is 0:0
Journal created on VMS platform
Type is Normal (unoptimized)
Open mode is Initial
Journal was backed up on19−JUL−2010 14:35:49.15
Backup type is No−Quiet−Point
I/O format is Block
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.4.5 RMU/COLLECT OPTIMIZER Invalid Cardinality With Vertical Record Partitioning

When the Oracle Rdb RMU/COLLECT OPTIMIZER command collected table cardinality statistics for tables with VERTICAL RECORD PARTITIONING, where table column values are assigned to different storage areas using a storage map, the cardinality was incorrectly incremented for each column partition. This caused the cardinality statistic for that table to be too large and also caused any statistics calculated using that cardinality value to be incorrect, such as the Row Clustering Factor. This has been fixed and the cardinality and related statistics will now be correct for tables with vertical record partitioning.

The following shows an example of the problem. The table T1 in the Rdb FOO database is defined with vertical record (column) partitioning. When the RMU/COLLECT OPTIMIZER command was run to collect statistics for the database, an incorrect table cardinality (5 instead of 1) and an incorrect table row clustering factor (3 instead of 15) were calculated for table T1. The row clustering factor was incorrect because it was calculated based on the incorrect table cardinality.

```
$ SQL
        CREATE DATA FILE FOO
                CREATE STORAGE AREA A1 FILE A1
                CREATE STORAGE AREA A2 FILE A2
                CREATE STORAGE AREA A3 FILE A3
                CREATE STORAGE AREA A4 FILE A4
                CREATE STORAGE AREA A5 FILE A5;
        CREATE TABLE T1 (C1 INT, C2 INT, C3 INT, C4 INT, C5 INT);
        CREATE STORAGE MAP M1 FOR T1
                STORE COLUMNS (C1) IN A1
                STORE COLUMNS (C2) IN A2
                STORE COLUMNS (C3) IN A3
                STORE COLUMNS (C4) IN A4
                STORE COLUMNS (C5) IN A5;
```

```
        INSERT INTO T1 VALUES (1,2,3,4,5);
1 row inserted
        COMMIT;
        EXIT;
$ RMU/COLLECT OPTIMIZER FOO.RDB
Start loading tables... at  8-SEP-2010 11:44:28.31
Done loading tables.... at  8-SEP-2010 11:44:28.33
Start loading indexes... at  8-SEP-2010 11:44:28.33
Done loading indexes.... at  8-SEP-2010 11:44:28.33
Start collecting btree index stats... at  8-SEP-2010 11:44:28.37
Done collecting btree index stats.... at  8-SEP-2010 11:44:28.37
Start collecting table & hash index stats... at  8-SEP-2010 11:44:28.37
Done collecting table & hash index stats.... at  8-SEP-2010 11:44:28.37
Start collecting workload stats... at  8-SEP-2010 11:44:28.45
Maximum memory required (bytes) = 0
Done collecting workload stats.... at  8-SEP-2010 11:44:28.45
Start calculating stats... at  8-SEP-2010 11:44:28.46
Done calculating stats.... at  8-SEP-2010 11:44:28.46
Start writing stats... at  8-SEP-2010 11:44:28.49

--------------------------------------------------------------------------------

Optimizer Statistics collected for table : T1

  Cardinality            : 5
  Row clustering factor  : 3.0000000
Done writing stats.... at  8-SEP-2010 11:44:28.51
```

The following example shows that this problem has been fixed. The table T1 in the Rdb FOO database is defined with vertical record (column) partitioning. When the RMU/COLLECT OPTIMIZER command is run to collect statistics for the database, a correct table cardinality (1) and a correct table row clustering factor (15) are calculated for table T1.

```
$ SQL
        CREATE DATA FILE FOO
                CREATE STORAGE AREA A1 FILE A1
                CREATE STORAGE AREA A2 FILE A2
                CREATE STORAGE AREA A3 FILE A3
                CREATE STORAGE AREA A4 FILE A4
                CREATE STORAGE AREA A5 FILE A5;
        CREATE TABLE T1 (C1 INT, C2 INT, C3 INT, C4 INT, C5 INT);
        CREATE STORAGE MAP M1 FOR T1
                STORE COLUMNS (C1) IN A1
                STORE COLUMNS (C2) IN A2
                STORE COLUMNS (C3) IN A3
                STORE COLUMNS (C4) IN A4
                STORE COLUMNS (C5) IN A5;
        INSERT INTO T1 VALUES (1,2,3,4,5);
1 row inserted
        COMMIT;
        EXIT;
$ RMU/COLLECT OPTIMIZER FOO.RDB
Start loading tables... at  8-SEP-2010 11:45:16.63
Done loading tables.... at  8-SEP-2010 11:45:16.69
Start loading indexes... at  8-SEP-2010 11:45:16.69
Done loading indexes.... at  8-SEP-2010 11:45:16.69
Start collecting btree index stats... at  8-SEP-2010 11:45:16.89
Done collecting btree index stats.... at  8-SEP-2010 11:45:16.89
Start collecting table & hash index stats... at  8-SEP-2010 11:45:16.89
Done collecting table & hash index stats.... at  8-SEP-2010 11:45:16.90
Start collecting workload stats... at  8-SEP-2010 11:45:17.01
```

```
Maximum memory required (bytes) = 0
Done collecting workload stats.... at  8-SEP-2010 11:45:17.02
Start calculating stats... at  8-SEP-2010 11:45:17.02
Done calculating stats.... at  8-SEP-2010 11:45:17.02
Start writing stats... at  8-SEP-2010 11:45:17.06

--------------------------------------------------------------------------------

Optimizer Statistics collected for table : T1

  Cardinality            : 1
  Row clustering factor  : 15.0000000
Done writing stats.... at  8-SEP-2010 11:45:17.07
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.4.6 RMU /RECOVER /ORDER_AIJ May Remove Required Journal Files

Bug 10020166

In certain cases, it was possible for the /ORDER_AIJ qualifier on the RMU/RECOVER command to reject (or "prune") required journals from the input journal list. This would most likely happen when all the following occurred:

- Performed an RMU /RECOVER using After−Image Journal (AIJ) backups;
- Recovered only a subset of the required journals;
- The last journal applied had unresolved or incomplete transactions;

For example, suppose a database needed to be recovered, starting at AIJ VNO 10 using the following AIJ backup journal files:

```
J1.AIJ (aij quiet_point backup)   : OPEN_VNO=10, QUIET_VNO=10
J2.AIJ (aij noquiet_point backup) : OPEN_VNO=11, QUIET_VNO=10
J3.AIJ (aij noquiet_point backup) : OPEN_VNO=12, QUIET_VNO=11
J4.AIJ (aij quiet_point backup)   : OPEN_VNO=13, QUIET_VNO=13
```

STEP 1) The DBA issues:

```
$ RMU/RECOVER/ORDER_AIJ/ROOT=MF_PERSONNEL J1,J2
```

Journals J1.AIJ (OPEN_VNO=10) and J2.AIJ (OPEN_VNO=11) are applied and the database root is updated to show that RCVR_VNO 12 is the next journal to be applied. However, J2 had potentially unresolved transactions since it was not closed at a quiet_point.

STEP 2) The DBA next issues:

```
$ RMU/RECOVER/ORDER_AIJ/ROOT=MF_PERSONNEL J3,J4
```

Since J3 (OPEN_VNO=12, QUIET_VNO=11) does not start at a quiet_point, recovery terminates with an error:

```
RMU-F-AIJNORCVR, recovery must start with journal sequence 11
```

STEP 3) The DBA next issues:

```
$ RMU/RECOVER/ORDER_AIJ/ROOT=MF_PERSONNEL J1,J2,J3,J4
```

Normally, this would have been the optimal command for the recovery in this case but since the root had been previously updated to show that VNO 12 is required to start recovery, /ORDER_AIJ would prune J1 and J2 and only attempt to recover journals J3 and J4, which would result in the same error as in STEP 2 above.

The workaround would be to not use /ORDER_AIJ when recovering with such a strategy.

This problem has been corrected in Oracle Rdb Release 7.2.5. Now when the DBA attempts STEP 1, the root's OPEN_VNO will not be updated past the last known QUIET_POINT (VNO 10, in this case).

Note

When using AIJ backups for recovery, RMU/RECOVER must begin with a QUIET_POINT backup. Oracle recommends that recovery terminate with an AIJ backup that was closed at a QUIET_POINT (i.e. the next journal backed up was created by a QUIET_POINT backup).

3.4.7 RMU/CONVERT Fails to Convert Databases With Database−wide Collating Sequence

Bug 10125553

In prior versions of Oracle Rdb, it was possible that databases created with a database−wide collating sequence would not be converted correctly to Oracle Rdb V7.2.

The following example shows the problem where the database was created using the following definition:

```
create database
     filename BAD
     protection is ACL
     collating sequence GERMAN
;
disconnect all;
```

The RMU/CONVERT would proceed in a similar manner to the following example.

```
$ RMU/CONVERT BAD /NOCONFIRM
%RMU−I−RMUTXT_000, Executing RMU for Oracle Rdb V7.2−411 on OpenVMS Alpha
V8.3−1H1
%RMU−I−LOGCONVRT, database root converted to current structure level
%RMU−S−CVTDBSUC, database USER1:[TESTING]BAD.RDB;1 successfully converted from
version V7.1 to V7.2
%RMU−I−CVTCOMSUC, CONVERT committed for USER1:[TESTING]BAD.RDB;1 to version V7.2
%RDB−E−OBSOLETE_METADA, request references metadata objects that no longer exist
−RDMS−F−RELNOEXI, relation RDB$DATABASE does not exist in this database
%RDB−E−OBSOLETE_METADA, request references metadata objects that no longer exist
−RDMS−F−RELNOEXI, relation RDB$DATABASE does not exist in this database
```

If this occurs, Oracle recommends restoring the database from backup and using SQL EXPORT DATABASE and IMPORT DATABASE under Rdb V7.2, or installing this corrected version prior to performing RMU/CONVERT.

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.4.8 RMU/CONVERT/NOCOMMIT Did Not Call "Fix Up" Routine at End of Conversion

Bug 10145825

The Oracle Rdb RMU/CONVERT/NOCOMMIT command converts an Rdb database from an older version of Rdb to a newer version of Rdb but retains the original metadata as well as the converted metadata so that later the user can either execute the RMU/CONVERT/COMMIT command to finalize the conversion by deleting the original metadata or return the database metadata to the original version of Rdb by executing the RMU/CONVERT/ROLLBACK command.

However, there was a problem where RMU/CONVERT/NOCOMMIT did not attach to the database once it had been converted and call a routine to fully complete the metadata conversion. This routine has always been called by RMU/CONVERT/COMMIT. This problem has been fixed and now RMU/CONVERT/NOCOMMIT will also attach to the database and call this routine to fully complete the metadata conversion.

Because RMU/CONVERT/NOCOMMIT did not call a routine to fully complete the metadata conversion, the following problems could occur.

- For Multischema databases, access to new system tables could fail because the new system tables were not added to the schema mapping table.
- Constraint type information, which would allow easier database queries and future query optimizations, would not be propagated from RDB$RELATION_CONSTRAINTS to RDB$CONSTRAINTS.
- If the database was used for Replication Option for Rdb, an index might not be added to improve performance (RDB$TRAN_RELS_REL_NAME_NDX on table RDB$TRANSFER_RELATIONS).
- A file level ACL might not be added or modified to enable RMU access to the database.

The following shows an example of the problem. The Rdb logical RDMS$SET_FLAGS is defined to see if an attach to the database is being done at the end of the conversion to call the routine to finalize the metadata conversion. The output shows that no attach to the database is being executed.

```
$ DEFINE RDMS$SET_FLAGS "DATABASE"
$ RMU/CONVERT/NOCOMMIT MF_PERSONNEL
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.2-410 on OpenVMS
 Alpha V8.4
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 V7.1 to V7.2
```

The following example shows that this problem has been fixed. Again, the Rdb logical RDMS$SET_FLAGS is defined to see if an attach to the database is being done at the end of the conversion to call the routine to finalize the metadata conversion. The output now shows that the attach to the database is being executed.

```
$ DEFINE RDMS$SET_FLAGS "DATABASE"
$ RMU/CONVERT/NOCOMMIT MF_PERSONNEL
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.2-501 on OpenVMS
 Alpha V8.4
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 V7.1 to V7.2
 ATTACH #1, Database
 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
 ~P Database Parameter Buffer (version=2, len=62)
 0000 (00000) RDB$K_DPB_VERSION2
 0001 (00001) RDB$K_FACILITY_ALL
 0002 (00002) RDB$K_DPB2_IMAGE_NAME
 "NODE::DEVICE:[DIRECTORY]RMU72.EXE"
 003A (00058) RDB$K_FACILITY_RDB_VMS
 003B (00059) RDB$K_DPB2_CHECK_ACCESS
 RDMS$BIND_WORK_FILE =
  "DEVICE:[DIRECTORY]RDMSTTBL$UN71RK1V4IA10EJ17T80.TMP;"
 (Visible = 0)
 DETACH #1
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.4.9 Problems Validating Files Specified in the "/AIJ_OPTIONS" File

Bug 10327835

Insufficient user privileges or invalid file device and directory specifications cause fatal file creation errors for AIJ related files contained in the options file specified by the "/AIJ_OPTIONS" qualifier used with the Oracle Rdb RMU "RMU/RESTORE", "RMU/RESTORE/ONLY_ROOT", "RMU/MOVE_AREA" and "RMU/COPY_DATABASE" commands.

However, these fatal file creation problems were not detected at the start of command execution when the options file specified by the "/AIJ_OPTIONS" qualifier is parsed but at the end of command execution at the time the AIJ related files specified in the "/AIJ_OPTIONS" file are actually created. Therefore, even though a fatal error was finally put out and the command was aborted, it was possible for an accessible Rdb database root file to be created pointing to the wrong AIJ file specification from the original Rdb database root instead of the AIJ file specification specified in the "/AIJ_OPTIONS" file.

This problem has been corrected in Oracle Rdb Release 7.2.5. Now these fatal AIJ file and AIJ backup file creation problems are detected at the start of the RMU command execution, immediately after the AIJ options file is parsed, so no invalid database files are created and a true abort of the command execution occurs.

The following example shows the problem. The device and directory "DEVICE:[DIRECTORY.TEST.BOGUS]", specified for the three fixed AIJ files in the options file

"bad_aij.opt" used with the "RMU/RESTORE" command, does not exist. However, this problem was not detected and the restore command was not aborted when the "bad_aij.opt" file was parsed at the start of the command. Instead, the problem was detected and the command was aborted at the end of the command, after all the other database files were restored, at the creation time of the AIJ files specified by the "bad_aij.opt" options file.

```
$ RMU/RESTORE/LOG/NOCDD/AIJ_OPTIONS=BAD_AIJ.OPT -
 /DIRECTORY=DEVICE:[DIRECTORY.TEST.TEST2] -
 /ROOT=DEVICE:[DIRECTORY.TEST.TEST2] MFP.RBF
%RMU-I-RESTXT_18, Processing options file BAD_AIJ.OPT
  JOURNAL IS ENABLED -
      RESERVE 6 -
      ALLOCATION IS 512 -
      EXTENT IS 512 -
      OVERWRITE IS DISABLED -
      SHUTDOWN_TIMEOUT IS 60 -
      NOTIFY IS DISABLED -
      BACKUPS ARE MANUAL -
      CACHE IS DISABLED
  ADD JOURNAL AIJ1 -
   FILE DEVICE:[DIRECTORY.TEST.BOGUS]AIJ1.AIJ;1
  !    FILE AIJ1
  ADD JOURNAL AIJ2 -
   FILE DEVICE:[DIRECTORY.TEST.BOGUS]AIJ2.AIJ;1
  !    FILE AIJ2
  ADD JOURNAL AIJ3 -
   FILE DEVICE:[DIRECTORY.TEST.BOGUS]AIJ3.AIJ;1
  !    FILE AIJ3
%RMU-I-RESTXT_00, Restored root file DEVICE:[DIRECTORY.TEST.TEST2]
MF_PERSONNEL.RDB;1
%RMU-I-RESTXT_21, Starting full restore of storage area (RDB$SYSTEM)
 DEVICE:[DIRECTORY.TEST.TEST2]MF_PERS_DEFAULT.RDA;1 at  4-FEB-2011 12:57:09.20
%RMU-I-RESTXT_21, Starting full restore of storage area (EMPIDS_OVER)
 DEVICE:[DIRECTORY.TEST.TEST2]EMPIDS_OVER.RDA;1 at  4-FEB-2011 12:57:09.21
%RMU-I-RESTXT_24, Completed full restore of storage area (EMPIDS_OVER)
 DEVICE:[DIRECTORY.TEST.TEST2]EMPIDS_OVER.RDA;1 at  4-FEB-2011 12:57:09.22
%RMU-I-RESTXT_24, Completed full restore of storage area (RDB$SYSTEM)
 DEVICE:[DIRECTORY.TEST.TEST2]MF_PERS_DEFAULT.RDA;1 at  4-FEB-2011 12:57:09.44
%RMU-I-RESTXT_01, Initialized snapshot file
 DEVICE:[DIRECTORY.TEST.TEST2]MF_PERS_DEFAULT.SNP;1
%RMU-I-LOGINIFIL,     contains 146 pages, each page is 4 blocks long
%RMU-I-RESTXT_01, Initialized snapshot file
 DEVICE:[DIRECTORY.TEST.TEST2]EMPIDS_OVER.SNP;1
%RMU-I-LOGINIFIL,     contains 10 pages, each page is 4 blocks long
%RMU-I-AIJWASON, AIJ journaling was active when the database was backed up
%RMU-I-AIJRECFUL, Recovery of the entire database starts with
 AIJ file sequence 0
%RMU-F-FILACCERR, error creating after-image journal file
 DEVICE:[DIRECTORY.TEST.BOGUS]AIJ1.AIJ;1
-RMS-E-DNF, directory not found
-SYSTEM-W-NOSUCHFILE, no such file
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at  4-FEB-2011 12:57:09.57
```

The following example shows that this problem has been fixed. Again, the device and directory "DEVICE:[DIRECTORY.TEST.BOGUS]", specified for the three fixed AIJ files in the options file "bad_aij.opt" used with the "RMU/RESTORE" command, does not exist. However, now the problem is detected and the restore command is aborted when the "bad_aij.opt" file is parsed at the start of the command, before any database files have been created.

```
$ RMU/RESTORE/LOG/NOCDD/AIJ_OPTIONS=BAD_AIJ.OPT -
 /DIRECTORY=DEVICE:[DIRECTORY.TEST.TEST2] -
 /ROOT=DEVICE:[DIRECTORY.TEST.TEST2] MFP.RBF
%RMU-I-RESTXT_18, Processing options file BAD_AIJ.OPT
  JOURNAL IS ENABLED -
      RESERVE 6 -
      ALLOCATION IS 512 -
      EXTENT IS 512 -
      OVERWRITE IS DISABLED -
      SHUTDOWN_TIMEOUT IS 60 -
      NOTIFY IS DISABLED -
      BACKUPS ARE MANUAL -
      CACHE IS DISABLED
  ADD JOURNAL AIJ1 -
   FILE DEVICE:[DIRECTORY.TEST.BOGUS]AIJ1.AIJ;1
  !    FILE AIJ1
  ADD JOURNAL AIJ2 -
   FILE DEVICE:[DIRECTORY.TEST.BOGUS]AIJ2.AIJ;1
  !    FILE AIJ2
  ADD JOURNAL AIJ3 -
   FILE DEVICE:[DIRECTORY.TEST.BOGUS]AIJ3.AIJ;1
  !    FILE AIJ3
%RMU-F-FILACCERR, error creating after-image journal file
 DEVICE:[DIRECTORY.TEST.BOGUS]AIJ1.AIJ;1
-RMS-E-DNF, directory not found
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at  4-FEB-2011
12:59:17.33
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.4.10 RMU Online Backup May Store TSNs of Zero

Bug 11769886

When an online RMU Backup operation captures data from a live page that requires searching a snapshot chain to retrieve visible row content, the TSN values written to the backup output would incorrectly be stored as zeros.

In certain cases, a database restored from such a backup may contain incorrect record content and may lead to verification or runtime errors.

This problem has been corrected in Oracle Rdb Release 7.2.5. Correct TSN values are now written to the backup output.

3.4.11 RMU/SET AFTER/AIJ_OPTIONS RMU-F-VALLSMIN Error If "RESERVE 0"

Bug 11741193

The Oracle Rdb RMU "RMU/SET AFTER/AIJ_OPTIONS" command failed with the error "RMU-F-VALLSSMIN, value (0) is less than minimum allowed value (1) for RESERVE" if "RESERVE 0" was specified or no "RESERVE" clause was specified in the AIJ options file. Therefore, even if sufficient AIJ file reservations were already defined for the database, there was no way to avoid reserving additional unneeded space in the after-image journal configuration when the "ADD JOURNAL" clause was used in the

AIJ options file specified by the "RMU/SET AFTER/AIJ_OPTIONS" command.

If the AIJ options file was not used and "RMU/SET AFTER/RESERVE=0" was specified on the command line, or the "/RESERVE" qualifier was not specified on the command line, then RMU/SET AFTER correctly did not reserve additional space in the after−image journal configuration. This is the way the RESERVE clause is supposed to work as specified in the Orace Rdb RMU documentation for the "RMU/SET AFTER" command. If the AIJ file reservations already defined for the database are not sufficient to hold the additional AIJ files added to the after−image journal file configuration by the "ADD JOURNAL" clause, the error "%RMU−F−NOAIJSLOTS, no more after−image journal slots are available" will continue to be output and the "RMU/SET AFTER" command will fail with no changes made to the database AIJ configuration.

This problem has been fixed. Now "RESERVE 0" can be specified in the AIJ options file used with the "RMU/SET AFTER" command or no "RESERVE" clause can be specified. In both cases, no additional space in the after−image journal configuration will be reserved.

The following examples show the problem. Both when no "RESERVE" clause is specified in the AIJ options file and when "RESERVE 0" is specified to indicate that no additional space in the after−image journal configuration should be reserved, the "RMU/SET AFTER" command incorrectly fails with the "%RMU−F−VALLSSMIN" error. Only when a reserve clause with a value greater than zero is specified does the "RMU/SET AFTER" command succeed.

```
$ type aij.opt
JOURNAL IS ENABLED
ADD JOURNAL AIJ1 FILE DEVICE:[DIRECTORY]AIJ1
$ RMU/SET AFTER /AIJ_OPTION=AIJ.OPT MF_PERSONNEL
%RMU−F−VALLSSMIN, value (0) is less than minimum allowed value (1)
 for RESERVE
%RMU−F−FTL_SET, Fatal error for SET operation at 10−FEB−2011
 08:11:46.75
$ type aij.opt
JOURNAL IS ENABLED RESERVE 0
ADD JOURNAL AIJ1 FILE DEVICE:[DIRECTORY]AIJ1
$ RMU/SET AFTER /AIJ_OPTION=AIJ.OPT MF_PERSONNEL
%RMU−F−VALLSSMIN, value (0) is less than minimum allowed value (1)
 for RESERVE
%RMU−F−FTL_SET, Fatal error for SET operation at 10−FEB−2011
 08:11:46.75
$ RMU/SET AFTER /AIJ_OPTION=AIJ.OPT MF_PERSONNEL
%RMU−I−RESTXT_18, Processing options file AIJ.OPT
  JOURNAL IS ENABLED RESERVE 1
  ADD JOURNAL AIJ1 FILE DEVICE:[DIRECTORY]AIJ1
%RMU−I−LOGMODFLG,     disabled after−image journaling
%RMU−I−LOGMODVAL,     reserved 1 additional after−image journals
%RMU−W−DOFULLBCK, full database backup should be done to ensure
 future recovery
%RMU−I−LOGCREAIJ, created after−image journal file
 DEVICE:[DIRECTORY]AIJ1.AIJ;1
%RMU−I−LOGMODSTR,     activated after−image journal "AIJ1"
%RMU−I−LOGMODFLG,     enabled after−image journaling
%RMU−W−DOFULLBCK, full database backup should be done to ensure
 future recovery
```

The following example shows that this problem has been fixed. Both when no "RESERVE" clause is specified in the AIJ options file and when "RESERVE 0" is specified to indicate that no additional space in the after−image journal configuration should be reserved the "RMU/SET AFTER" command succeeds and the "%RMU−I−LOGMODVAL, reserved 1 additional after−image journals" file message does not come out

indicating that no additional space in the AIJ configuration has been reserved but currently available space has been used.

```
$ RMU/SET AFTER /AIJ_OPTION=AIJ.OPT MF_PERSONNEL
%RMU-I-RESTXT_18, Processing options file AIJ.OPT
  JOURNAL IS ENABLED
  ADD JOURNAL AIJ1 FILE DEVICE:[DIRECTORY]AIJ1
%RMU-I-LOGMODFLG,     disabled after-image journaling
%RMU-W-DOFULLBCK, full database backup should be done to ensure
 future recovery
%RMU-I-LOGCREAIJ, created after-image journal file
 DEVICE:[DIRECTORY]AIJ1.AIJ;1
%RMU-I-LOGMODSTR,     activated after-image journal "AIJ1"
%RMU-I-LOGMODFLG,     enabled after-image journaling
%RMU-W-DOFULLBCK, full database backup should be done to ensure
 future recovery
$ RMU/SET AFTER /AIJ_OPTION=AIJ.OPT MF_PERSONNEL
%RMU-I-RESTXT_18, Processing options file AIJ.OPT
  JOURNAL IS ENABLED RESERVE 0
  ADD JOURNAL AIJ1 FILE DEVICE:[DIRECTORY]AIJ1
%RMU-I-LOGMODFLG,     disabled after-image journaling
%RMU-W-DOFULLBCK, full database backup should be done to ensure
 future recovery
%RMU-I-LOGCREAIJ, created after-image journal file
 DEVICE:[DIRECTORY]AIJ1.AIJ;1
%RMU-I-LOGMODSTR,     activated after-image journal "AIJ1"
%RMU-I-LOGMODFLG,     enabled after-image journaling
%RMU-W-DOFULLBCK, full database backup should be done to ensure
 future recovery
```

As indicated above, to avoid this problem in a previous version of Oracle Rdb RMU, either specify a non−zero reserve clause in the AIJ options file or do not use the AIJ options file with the "RMU/SET AFTER" command but use the "/RESERVE" and "/ADD" qualifiers on the command line instead.

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.4.12 RMU/BACKUP/PARALLEL/RESTORE_OPTIONS Was Not Fully Supported

Bug 6767004

The "/RESTORE_OPTIONS" qualifier, when used with the Oracle Rdb RMU "RMU/BACKUP" command, specifies the file specification of an options file that the backup command creates which contains storage area names followed by storage area qualifiers that define the attributes of the storages areas in the database being backed up. This options file can then be edited and specified by the "/OPTIONS" qualifier used with the "RMU/RESTORE" command to change the attributes of these storage areas when the database is restored.

The "/RESTORE_OPTIONS" qualifier did not work properly when used with the "/PARALLEL" qualifier on the "RMU/BACKUP" command line. The "RESTORE_OPTIONS" option was not put in the PLAN file and was therefore never passed to the "Coordinator" process for execution in a multi−process environment. Therefore, if the "RMU/BACKUP/PLAN" command was used to do a parallel backup using the PLAN file, or when the default "/EXECUTE" qualifier was specified with the "RMU/BACKUP/PARALLEL" command, no restore options file was created.

This problem has been fixed. Now when the "/RESTORE_OPTIONS" qualifier is specified on the "RMU/BACKUP/PARALLEL" command line, the entry

```
       Restore_options = file
```

will be added to the PLAN file, where "file" is the valid VMS file specification of the options file specified by the "/RESTORE_OPTIONS = file" qualifier on the command line, and the options file will be created when the PLAN file is executed. If the "/RESTORE_OPTIONS" qualifier is not specified on the command line, the following "place holder" entry in the form of a comment will be put in the PLAN file and no restore options file will be created when the PLAN file is executed.

```
      ! Restore_options = Restore options file
```

The following example shows the problem. When the parallel backup is executed, the "TMP.PLAN" file created does not contain the "RESTORE_OPTIONS" option and the "RES.OPT" restore options file is not created. When the parallel backup is repeated based on the same "TMP.PLAN" file, the "RES.OPT" file is again not created.

```
$ RMU/BACKUP/PARALLEL=EXEC=1/DISK=WRITER=2/THREADS=3-
 /RESTORE_OPTIONS=DEVICE:[DIRCTORY]RES.OPT/EXEC/LIST=TMP.PLAN-
  MF_PERSONNEL [.DIR1]MFP,[.DIR2]
%RMU-I-COMPLETED, BACKUP operation completed at
 2-MAR-2011 15:19:19.40
$ SEAR TMP.PLAN RESTORE_OPTIONS
%SEARCH-I-NOMATCHES, no strings matched
$ DIR *.OPT
%DIRECT-W-NOFILES, no files found
$ RMU/BACKUP/PLAN TMP.PLAN
%RMU-I-COMPLETED, BACKUP operation completed at
 2-MAR-2011 15:30:19.40
$ DIR *.OPT
%DIRECT-W-NOFILES, no files found
```

The following example shows that this problem has been fixed. When the parallel backup is executed, the "TMP.PLAN" file created does contain the "RESTORE_OPTIONS" option and the "RES.OPT" restore options file is created. When the parallel backup is repeated based on the same "TMP.PLAN" file, the "RES.OPT" file is again created.

```
$ RMU/BACKUP/PARALLEL=EXEC=1/DISK=WRITER=2/THREADS=3-
 /RESTORE_OPTIONS=DEVICE:[DIRCTORY]RES.OPT/EXEC/LIST=TMP.PLAN-
  MF_PERSONNEL [.DIR1]MFP,[.DIR2]
%RMU-I-COMPLETED, BACKUP operation completed at
 3-MAR-2011 15:19:19.40
$ SEAR TMP.PLAN RESTORE_OPTIONS
    Restore_options = DEVICE:[DIRECTORY]RES.OPT
$ DIR *.OPT

Directory DEVICE:[DIRECTORY]

RES.OPT;1
$ RMU/BACKUP/PLAN TMP.PLAN
%RMU-I-COMPLETED, BACKUP operation completed at
 3-MAR-2011 15:30:19.40
$ DIR *.OPT

Directory DEVICE:[DIRECTORY]
```

```
RES.OPT;2
```

This problem has been corrected in Oracle Rdb Release 7.2.5.

3.5 LogMiner Errors Fixed

3.5.1 RMU/UNLOAD/AFTER_JOURNAL /STATISTICS With /OUTPUT Information Display

In prior versions of Oracle Rdb, when using the "/STATISTICS" and "/OUTPUT" qualifiers with the RMU/UNLOAD/AFTER_JOURNAL command, the periodic statistics display records (containing the elapsed time, CPU time, IO counts and so on) would be written to SYS$OUTPUT rather than to the output file as specified.

This problem has been corrected in Oracle Rdb Release 7.2.5. The periodic statistics display record is now written to the output file rather than to SYS$OUTPUT.

3.6 Row Cache Errors Fixed

3.6.1 Row Caching Remains Unexpectedly Disabled for a Newly Added Storage Area

Bug 5926180

In prior releases, a DROP STORAGE AREA for an area that was cached would cause row caching for that cache to become disabled. Subsequently, when a new area was added which also referenced that same cache, it remained disabled. An ALTER DATABASE ... ALTER STORAGE AREA ... CACHE USING was required to reenable row caching for that cache.

The following example shows that after the ADD STORAGE AREA, row caching for the area remains disabled.

```
SQL> alter database
cont>     filename KOD_ADD_AREA_CACHE_USING_7_DB
cont>     drop storage area KOD_ADD_AREA_CACHE_USING_7_AREA
cont> ;
SQL> $ rmu/dump/header KOD_ADD_AREA_CACHE_USING_7_DB/out=middle.txt
SQL> $ search middle.txt "Row Caching..."/wind=(0,2)
    Row Caching...
      - Row caching is enabled
      - No row cache is defined for this area

SQL> alter database
cont>     filename KOD_ADD_AREA_CACHE_USING_7_DB
cont>     add storage area KOD_ADD_AREA_CACHE_USING_7_AREA
cont>         page format is MIXED
cont>         cache using BIG_CACHE
cont> ;
SQL> $ rmu/dump/header KOD_ADD_AREA_CACHE_USING_7_DB/out=after.txt
SQL> $ search after.txt "Row Caching..."/wind=(0,2)
    Row Caching...
      - Row caching is enabled
      - No row cache is defined for this area
***************
    Row Caching...
      - Row caching is disabled
      - Row cache ID is 1

SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.5. The ADD STORAGE AREA command now implicitly enables row caching for the cache specified in the CACHE USING clause.

3.7 RMU Show Statistics Errors Fixed

3.7.1 Stall Statistics (Aggregate Count) In RMU /SHOW STATISTICS Inaccurate

Bug 10016136

In previous versions of Oracle Rdb, the values on the "Stall Statistics (Aggregate Count)" display were incorrectly scaled and were displayed as inaccurate values.

This problem has been corrected in Oracle Rdb Release 7.2.5. The correct scaling factor is now used.

3.7.2 Unexpected ACCVIO When Using RMU/SHOW STATISTICS

Bug 11871974

In prior releases of Oracle Rdb, it was possible to receive an ACCVIO as RMU Show Statistics was returning to DCL. This problem was most likely due to an error freeing virtual memory used during the session. The ACCVIO causes a bugcheck dump to be generated with a footprint similar to the following.

- Itanium OpenVMS 8.3−1H1
- Oracle Rdb Server 7.2.4.1.0
- Got a RMUBUGCHK.DMP
- SYSTEM−F−ACCVIO, access violation, virtual address=FFFFFFFF816F4040
- Exception occurred at RMU72\COSI_MEM_FREE_VMLIST + 00000132
- Called from RMU72\KUT$DISPLAY + 00004DB0
- Called from RMU72\RMU$DISPLAY + 000044E0
- Called from RMU72\RMU$SHOW + 000014D0

This problem has been corrected in Oracle Rdb Release 7.2.5.

Chapter 4
Enhancements And Changes Provided in Oracle Rdb Release 7.2.5.1

4.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.5.1

4.1.1 New RMU Options File to Modify the Row Cache Backing Store Directories

The Oracle Rdb RMU "/SET ROW CACHE" command currently allows the Row Cache per−database default backing store directory and the database per−cache backing store directories to be specified or removed in the database root file. If the specified per−cache backing store directory is removed, the per−database backing store directory will be used and if the specified per−database backing store directory is removed, the root file device and directory will be the default backing store location. When the RMU/RESTORE, RMU/COPY_DATABASE and RMU/MOVE_AREA commands are used to change the location of the database root file, it would be convenient to be able to also modify the per−database and/or per−cache Row Cache backing store directories at the same time. This new feature allows an options file to be specified with the RMU/RESTORE, RMU/COPY_DATABASE and RMU/MOVE_AREA commands to modify and/or remove the per−database and/or per−cache Row Cache backing store directory specifications.

The following new optional qualifier can be specified with the RMU/RESTORE, RMU/COPY_DATABASE and RMU/MOVE_AREA commands to specify a Row Cache backing store options file which contains per−database and/or per−cache Row Cache backing store directory specifications.

```
/ROW_CACHE_OPTIONS = file_specification
```

The "file_specification" must be a valid file specification of an existing options file. This qualifier cannot be negated. The following syntax must be used in the row cache backing store options file.

```
$ type filename.opt
/BACKING_STORE = device:[directory]
row_cache_name /BACKING_STORE = device:[directory]
row_cache_name /NOBACKING_STORE
```

To modify or remove the current per−database default backing store location, either "/BACKING_STORE =" followed by a valid directory specification or "/NOBACKING_STORE" must be specified on the first line without being preceded by a Row Cache name. To modify or remove a current per−cache backing store location, an existing Row Cache name currently defined in the database root file must be specified followed by either "/BACKING_STORE =" followed by a valid directory specification or "/NOBACKING_STORE" must be specified. The wild card characters "%" and/or "*" can be specified as part of the Row Cache name, where "*" can be used in the place of one or more contiguous characters and "%" can be used in the place of a single character. In this case all matching per−cache backing store entries will be modified with the following "/BACKING_STORE =" or "/NOBACKING_STORE" specification. The RMU/DUMP/HEADER command can be used to display the current per−database default Row Cache backing store directory as well as the current per−cache Row Cache backing store directories.

For the RMU/RESTORE command, the /ROW_CACHE_OPTIONS qualifier cannot be used with the /AREA, /INCREMENTAL, /JUST_CORRUPT, /DUPLICATE or /ONLINE qualifiers. It can be used for a full or /ONLY_ROOT restore. For the RMU/MOVE_AREA command, the /ROW_CACHE_OPTIONS qualifier cannot be used with the /ONLINE or /QUIET_POINT qualifiers. It can only be used if the /ROOT qualifier is specified. For the RMU/COPY_DATABASE command, the /ROW_CACHE_OPTIONS qualifier

cannot be used with the /ONLINE, /QUIET_POINT or /DUPLICATE qualifiers.

The following example shows the RMU/RESTORE, RMU/MOVE_AREA, and RMU/COPY_DATABASE commands used with the /ROW_CACHE_OPTIONS qualifier to read backing store directory options files to modify or remove the current per−database and per−cache Row Cache backing store directories in the database root file. The contents of the options file read is displayed when the command is executed and the RMU/DUMP/HEADER command is used to check the modified backing store directories in the database root file.

```
$ SET VERIFY
$ RMU/RESTORE/NOCDD/NOLOG/DIR=TEST$DIRECTORY−
 /ROW_CACHE_OPTIONS=TEST$DIRECTORY:BSTORE.OPT −
 TEST$DIRECTORY:RSA.RBF
%RMU−I−RESTXT_18, Processing options file BSTORE.OPT
  /BACKING_STORE=DISK:[DIRECTORY]
  SAL_CACHE /BACKING_STORE=DISK:[DIRECTORY]
  JOB_CACHE/BACKING_STORE=DISK:[DIRECTORY]
  DROP_CACHE /BACKING_STORE=DISK:[DIRECTORY]

$ RMU/DUMP/HEADER/OUT=HDR.LIS TEST$DIRECTORY:RMU_SET_RCACHE_ALTER_DB
$ SEAR HDR.LIS "DEFAULT BACKING FILE DIRECTORY"
          Default backing file directory is "DISK:[DIRECTORY]"
$ SEAR HDR.LIS "CACHE FILE DIRECTORY"
      − Cache file directory is "DISK:[DIRECTORY]"
      − Cache file directory is "DISK:[DIRECTORY]"
      − Cache file directory is "DISK:[DIRECTORY]"
$ RMU/MOVE_AREA/NOLOG/DIRECTORY=DISK:[DIRECTORY]−
 /ROOT=DISK:[DIRECTORY]FILENAME.EXT
 /ROW_CACHE_OPTIONS=TEST$DIRECTORY:WBSTORE.OPT −
 TEST$DIRECTORY:RMU_SET_RCACHE_ALTER_DB
%RMU−I−RESTXT_18, Processing options file WBSTORE.OPT
  *CACHE /BACKING_STORE=DISK:[DIRECTORY]

$ RMU/DUMP/HEADER/OUT=HDR.LIS DISK:[DIRECTORY]FILENAME.EXT
$ SEAR HDR.LIS "DEFAULT BACKING FILE DIRECTORY"
          Default backing file directory is "DISK:[DIRECTORY]"
$ SEAR HDR.LIS "CACHE FILE DIRECTORY"
      − Cache file directory is "DISK:[DIRECTORY]"
      − Cache file directory is "DISK:[DIRECTORY]"
      − Cache file directory is "DISK:[DIRECTORY]"
$ RMU/COPY_DATABASE/NOLOG/DIRECTORY=DISK:[DIRECTORY]−
 /ROW_CACHE_OPTIONS=TEST$DIRECTORY:NOBSTORE.OPT −
 TEST$DIRECTORY:RMU_SET_RCACHE_ALTER_DB
%RMU−I−RESTXT_18, Processing options file NOBSTORE.OPT
  /NOBACKING_STORE
  SAL_CACHE /NOBACKING_STORE
  JOB_CACHE/NOBACKING_STORE
  DROP_CACHE /NOBACKING_STORE

$ RMU/DUMP/HEADER/OUT=HDR.LIS DISK:[DIRECTORY]FILENAME.EXT
$ SEAR HDR.LIS "DEFAULT BACKING FILE DIRECTORY"
          Default backing file directory is database directory
$ SEAR HDR.LIS "CACHE FILE DIRECTORY"
      − Derived cache file directory is "DISK:[DIRECTORY]"
      − Derived cache file directory is "DISK:[DIRECTORY]"
      − Derived cache file directory is "DISK:[DIRECTORY]"
```

4.1.2 New RMU/REPAIR Options File to Initialize Database Snapshot Files

The Oracle Rdb RMU "RMU/REPAIR/INITIALIZE=SNAPSHOTS" command currently allows the user to create and/or initialize new snapshot files when a snapshot file is deleted or corrupted. The "RMU/REPAIR/INITIALIZE=SNAPSHOTS=CONFIRM" command prompts the user not only to initialize snapshot files but also to optionally rename, move, or change the allocation of snapshot files. The "/AREAS" qualifier can also be specified to only initialize the snapshot files associated with the specified list of storage area names. This new feature allows the use of an options file with RMU/REPAIR to initialize snapshot files. This allows the user to avoid prompts for each snapshot file to be initialized and makes it more convenient to initialize a large number of options files in one RMU/REPAIR command.

The following optional new syntax can be specified with the RMU/REPAIR command to use an options file to initialize snapshot files.

```
/INITIALIZE=SNAPSHOTS=OPTIONS=file_specification
```

The "file_specification" must be the valid file specification of an existing options file. This qualifier cannot be negated. The following syntax must be used in the specified options file.

```
$ type filename.opt
storage_area [/SNAPSHOT=([FILE=file_specification][,][ALLOCATION=n])]
```

In the above, "storage_area" specifies the name of the storage area which uses the snapshot file. The "RMU/DUMP/HEADER" command can be used to display the storage area names contained in the database root file − see 'Snapshot area for storage area "storage_area"'. "FILE=file_specification" specifies an optional new file specification for the snapshot file. The "RMU/DUMP/HEADER" command can be used to display the snapshot file specifications contained in the database root file − see 'Filename is "device:[directory]name.SNP;1"'. The snapshot file specification cannot be changed for a single file database. "ALLOCATION=n" specifies an optional new page count allocation size for the snapshot file. By specifying a new allocation for a snapshot file, you can truncate a snapshot file or make it larger. The "RMU/DUMP/HEADER" command can be used to display the allocation size for the snapshot file − see "Current physical page count is n". Note that if the "/AREAS" qualifier is used in the same RMU/REPAIR command with "/INITIALIZE=SNAPSHOTS=OPTIONS=file_specification" it will be in effect for other RMU/REPAIR options but will be ignored for the "/INITIALIZE=SNAPSHOTS" option since the options file specifies the storage areas that use the snapshot files to be initialized.

In the following example, "SET VERIFY" is specified to display the contents of the JUST_AREA.OPT options file which specifies that the snapshot files for the departments, jobs and salary_history storage areas in the MF_PERSONNEL database are to be initialized. Since the JOBS.SNP snapshot file has been deleted, it will be created before it is initialized. The current snapshot file specifications and allocations are used. An RMU/VERIFY of MF_PERSONNEL is done after the repair to verify the integrity of the database.

```
$ set verify
$ del jobs.snp;*
$ dir jobs.snp
%DIRECT-W-NOFILES, no files found
$!
$ RMU/REPAIR/INIT=SNAPSHOTS=OPTIONS=JUST_AREA.OPT MF_PERSONNEL
%RMU-I-FULBACREQ, A full backup of this database should be
 performed after RMU REPAIR
  departments
```

```
  jobs
  salary_history
$!
$ dir jobs.snp

Directory DEVICE:[DIRECTORY]

JOBS.SNP;1

Total of 1 file.

$ RMU/VERIFY/ALL/NOLOG MF_PERSONNEL
$
```

In the following example, "SET VERIFY" is specified to display the contents of the AREA_SNAP.OPT options file which specifies that the snapshot files for the departments, jobs and salary_history storage areas in the MF_PERSONNEL database are to be initialized with a new file specification and page count allocation. An RMU/VERIFY of MF_PERSONNEL is done after the repair to verify the integrity of the database.

```
$ set verify
$ RMU/REPAIR/INIT=(SNAPSHOTS=OPTIONS=AREA_SNAP.OPT) MF_PERSONNEL
%RMU-I-FULBACREQ, A full backup of this database should be
 performed after RMU REPAIR
  departments /snapshot=(file=new_dept,allocation=200)
  jobs /snapshot=(file=new_jobs,allocation=210)
  salary_history /snapshot=(file=new_sal_hist,allocation=220)
$ RMU/VERIFY/ALL/NOLOG MF_PERSONNEL
$
```

4.1.3 RDMSTT Image Optionally Installed

Bug 3981803

If you plan on using the cluster capability of RMU/SHOW STATISTICS, Oracle recommends that you install the RDMSTT72.EXE image on OpenVMS with the appropriate privileges.

For Oracle Rdb Release 7.2.5.1 and later, the RMONSTART72.COM command procedure has been modified to optionally install the RDMSTT72.EXE image at monitor startup time. To take advantage of this, you will need to edit the RMONSTART72.COM procedure and remove the comment characters from the following lines:

```
$ ! DEFX SYS$COMMON:[SYSEXE]RDMSTT72.EXE
$ !     REMOVEX
$ !     ADDX /OPEN/HEAD/PROT/PRIV=(CMKRNL,SYSPRV,SHARE)
```

Also, edit the RMONSTOP72.COM procedure and remove the comment characters from the following lines:

```
$ ! DEFX SYS$COMMON:[SYSEXE]RDMSTT72.EXE
$ !     REMOVEX
```

This feature has been added to Oracle Rdb Release 7.2.5.1.

4.1.4 RMU Show Statistics Now Includes New Rdb Executive Statistics

This release of Oracle Rdb changes the RMU Show Statistics command in the following ways:

- The Rdb Executive Statistics screen now includes two new items: "request create" and "request release".
  These statistics closely correspond to the dynamic SQL DECLARE CURSOR and RELEASE statements. In general, interactive SQL will create and release a request for each interactive query, unless it is a cursor which will be retained until DISCONNECT. Compiled (SQL$MOD or SQL$PRE) applications will release queries only upon DISCONNECT (unless they are using dynamic SQL).
  The statistic "request create" counts the loading of a query, stored procedure, or stored function. The statistic "request release" counts the release of such requests (which will include release of virtual memory and release of metadata locks). Note that these statistics will also include queries executed by the SQL runtime environment.
- The statistic named "queries compiled" has been renamed as "subquery compiles" to better describe its function. Any configuration file which uses the old name will need to be changed. The new "request create" is a better metric if a count of compiled queries is required.
  This statistic is incremented when each table context is compiled, such as each branch of a UNION, join or nested subquery.

Chapter 5
Enhancements And Changes Provided in Oracle Rdb Release 7.2.5.0

5.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.5.0

5.1.1 RMU /SHOW STATISTICS /ROWS= and /COLUMNS= Feature

Previously, it was not possible to use the RMU /SHOW STATISTICS and specify the number of display rows and columns. The values would default to the user's display or, in the case of non−interactive mode, 132 columns and 66 rows.

This problem has been corrected in Oracle Rdb Release 7.2.5. RMU /SHOW STATISTICS includes the new qualifiers /ROWS=n and /COLUMNS=n to allow the user to specify the desired number of display rows and columns. The existing minimum and maximum limits apply as enforced by the SMG run time library or the RMU /SHOW STATISTICS utility.

5.1.2 New LIMIT Clauses Implemented for the CREATE and ALTER PROFILE Statement

In prior versions of Oracle Rdb, the LIMIT clauses of the CREATE PROFILE statement and the ALTER PROFILE statement were incomplete. These clauses are now active in this release of Rdb.

The following information replaces the description of these clauses in the current Oracle Rdb SQL Reference Manual, Volume 2 for the CREATE PROFILE Statement.

Arguments

- LIMIT CPU TIME
  LIMIT CPU TIME sets the maximum CPU time that can be used by the query compiler. The keyword DEFAULT indicates that no value is defined by this profile and is equivalent to NO LIMIT CPU TIME.
  If a numeric value or the keyword UNLIMITED is specified then this value will be used even when the SET QUERY LIMIT CPU TIME statement is present in the session, or when the logical name RDMS$BIND_QG_CPU_TIMEOUT is defined.
  NO LIMIT CPU TIME is the default. Units can be specified as seconds or minutes.
- LIMIT TIME
  LIMIT TIME sets the maximum elapsed time that can be used by the query compiler. The keyword DEFAULT indicates that no value is defined by this profile and is equivalent to NO LIMIT TIME.
  If a numeric value or the keyword UNLIMITED is specified then this value will be used even when the SET QUERY LIMIT TIME statement is present in the session, or when the logical name RDMS$BIND_QG_TIMEOUT is defined.
  NO LIMIT TIME is the default. Units can be specified as seconds or minutes.
- LIMIT ROWS
  LIMIT ROWS sets the maximum number of rows that can be returned by a query started by the user. The keyword DEFAULT indicates that no value is defined by this profile and is equivalent to NO LIMIT ROWS.
  If a numeric value or the keyword UNLIMITED is specified then this value will be used even when

the SET QUERY LIMIT ROWS statement is present in the session, or when the logical name RDMS$BIND_QG_REC_LIMIT is defined.
NO LIMIT ROWS is the default.

Examples

This example shows the use of the LIMIT clauses to set boundaries for standard database users.

```
SQL> create profile STANDARD_USER
cont>    limit rows 10000
cont>    limit time 10 minutes
cont>    limit cpu time 20 seconds;
SQL> show profile STANDARD_USER;
     STANDARD_USER
     Limit rows 10000
     Limit time 10 minutes
     Limit CPU time 20 seconds
SQL> alter profile STANDARD_USER
cont>    limit time 60 minutes;
```

Usage Notes

- The logical names RDMS$BIND_QG_REC_LIMIT, RDMS$BIND_QG_TIMEOUT, and RDMS$BIND_QG_CPU_TIMEOUT establish the process defaults for the query limit.
- The command SET QUERY LIMIT establishes the session default (unless already set by the query governor logical names).
- The profile LIMIT will either use these established defaults (LIMIT ... DEFAULT or NO LIMIT) or override them (LIMIT ... UNLIMITED or specified value).

5.1.3 Use of RMS MBC Larger Than 127

This release of Oracle Rdb takes advantage of OpenVMS enhancements permitting values of the RMS Multi Block Count (MBC) parameter to be up to 255 blocks (the prior limit was 127 blocks). With this change, some disk−based file read and write operations performed by Oracle Rdb may require half of the IO resources as compared with prior releases by allowing RMS to do larger IO transfers.

Oracle Rdb now anticipates that OpenVMS patch(es) have been installed that support using an RMS Multi Block Count (MBC) parameter larger than 127 blocks. Oracle Rdb will first attempt to use a larger value and if an RMS$_MBC error is returned from the SYS$CONNECT call, a second attempt is made with a RMS Multi Block Count (MBC) parameter of less than 128.

In order to receive the IO performance improvements available when accessing sequential files when using an RMS Multi Block Count (MBC) parameter larger than 127 blocks, the following patches (or their subsequent replacements) are required to be installed:

- VMS84I_UPDATE−V0400 or later
- VMS84A_UPDATE−V0400 or later
- VMS831H1I_SYS−V1200 or later
- VMS83I_SYS−V1500 or later
- VMS83A_SYS−V1800 or later

Oracle recommends installing OpenVMS patches that permit values of the RMS Multi Block Count (MBC) parameter to be up to 255 blocks for best performance and functionality.

5.1.4 New Optimizations for the LIKE Predicate

Bugs 3516321, 9931047 and 9910624

This release of Oracle Rdb will try to rewrite the LIKE predicate when the LIKE pattern is a string literal. This enhancement allows Rdb to simplify some LIKE expressions resulting in reduced I/O and CPU time required to satisfy these queries.

- When the LIKE pattern is only the "%" character (which matches one or more characters in the source string) then this will be replaced with an OCTET_LENGTH (...) >= 0 condition that does not require any pattern matching. Note that a string of "%" wildcards, such as "%%%%%", behaves in the same way as a single "%" character. Therefore, the same optimization is applied when any number of trailing % wildcards are detected.

```
SQL> select count(*)
cont>  from employees
cont>  where middle_initial like '%';
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (*)
Conjunct: OCTET_LENGTH (0.MIDDLE_INITIAL) >= 0
Get     Retrieval sequentially of relation 0:EMPLOYEES

          64
1 row selected
SQL>
```

- When the LIKE pattern is only a series of "_" characters (one or more) and the pattern length is the same size as the CHAR source expression, or the VARCHAR source expression matches the length of the pattern, then this will be replaced with an OCTET_LENGTH (...) = n condition that does not require any pattern matching.
  The following example shows a LIKE match against a fixed length CHAR column STATUS_NAME.

```
SQL> select status_name
cont>  from work_status
cont>  where status_name like '________';
Tables:
  0 = WORK_STATUS
Conjunct: OCTET_LENGTH (0.STATUS_NAME) = 8
Get     Retrieval sequentially of relation 0:WORK_STATUS
 STATUS_NAME
 INACTIVE
 ACTIVE
 ACTIVE
3 rows selected
SQL>
```

  The following example shows the string matching the variable length VARCHAR value which is six octets in length.

```
SQL> -- Find six character LAST_NAME values
SQL> select first_name, middle_initial, last_name
```

```
cont>  from candidates
cont>  where last_name like '______';
Tables:
  0 = CANDIDATES
Conjunct: OCTET_LENGTH (0.LAST_NAME) = 6
Get     Retrieval sequentially of relation 0:CANDIDATES
 FIRST_NAME   MIDDLE_INITIAL   LAST_NAME
 Oscar        M.               Wilson
1 row selected
SQL>
```

- When the LIKE pattern starts with a string not containing wildcard characters ("%", "_", or program supplied escape character) but followed by only the "%" wildcard, then Oracle Rdb can replace this with the STARTING WITH operator.

```
SQL> select employee_id, first_name, last_name
cont>  from employees
cont>  where last_name like 'Smith%';
Tables:
  0 = EMPLOYEES
Leaf#01 FFirst 0:EMPLOYEES Card=100
  Bool: 0.LAST_NAME STARTING WITH 'Smith'
  BgrNdx1 EMPLOYEES_LAST_NAME [1:1] Fan=14
    Keys: 0.LAST_NAME STARTING WITH 'Smith'
 EMPLOYEE_ID   FIRST_NAME   LAST_NAME
 00165         Terry        Smith
 00209         Roger        Smith
2 rows selected
SQL>
```

Note

Oracle Rdb has for some time made use of pattern prefix strings to start index scans when possible. This feature is now highlighted in the STRATEGY output by displaying "(Starting With)" after the LIKE predicate for the index keys. This optimization is applied to string literals, column references, host variables, and other string expressions.

```
SQL> declare :patt varchar(10) = 'Smit%';
SQL> select employee_id, first_name, last_name
cont>  from employees
cont>  where last_name like :patt;
Tables:
  0 = EMPLOYEES
Leaf#01 FFirst 0:EMPLOYEES Card=100
  Bool: 0.LAST_NAME LIKE <var0>
  BgrNdx1 EMPL_LAST_NAME [1:1] Fan=12
    Keys: 0.LAST_NAME LIKE <var0> (Starting With)
    Bool: 0.LAST_NAME LIKE <var0>
 EMPLOYEE_ID   FIRST_NAME   LAST_NAME
 00165         Terry        Smith
 00209         Roger        Smith
2 rows selected
SQL>
```

Conversely, if the LIKE pattern string literal has leading wildcards, then this optimization will be disabled since we know during query compile that such an optimization would provide no benefit and thus we save CPU time for such queries.

```
SQL> select employee_id, first_name, last_name
cont>  from employees
cont>  where last_name like '%Smith%';
Tables:
  0 = EMPLOYEES
Leaf#01 FFirst 0:EMPLOYEES Card=100
  Bool: 0.LAST_NAME LIKE '%Smith%'
  BgrNdx1 EMPL_LAST_NAME [0:0] Fan=12
    Keys: 0.LAST_NAME LIKE '%Smith%'
    Bool: 0.LAST_NAME LIKE '%Smith%'
 EMPLOYEE_ID   FIRST_NAME   LAST_NAME
 00165         Terry        Smith
 00209         Roger        Smith
2 rows selected
SQL>
```

- When the LIKE pattern does not contain any wildcard characters, then the LIKE may be converted to an equals condition with a restriction on the length. In general, this produces better query strategies as it allows leading index segments to be matched, where in prior versions this LIKE reference would terminate the partial key construction.

```
SQL> select count (*)
cont>  from employees
cont>  where sex like 'F';
Tables:
  0 = EMPLOYEES
Aggregate: 0:COUNT (*)
Conjunct: 0.SEX = 'F'
Get     Retrieval by index of relation 0:EMPLOYEES
  Index name  EMPLOYEES_LAST_NAME [0:0]

                   35
1 row selected
SQL>
```

This example uses a CANDIDATES table where the LAST_NAME column is an indexed VARCHAR type. Observe that the LIKE was transformed to an equals condition that enabled a direct index lookup.

```
SQL> select last_name, first_name
cont> from CANDIDATES
cont>  where last_name like 'Wilson';
Tables:
  0 = CANDIDATES
Leaf#01 FFirst 0:CANDIDATES Card=3
  Bool: (0.LAST_NAME = 'Wilson') AND (OCTET_LENGTH (0.LAST_NAME) = 6)
  BgrNdx1 CAND_LAST_NAME [1:1] Fan=12
    Keys: 0.LAST_NAME = 'Wilson'
 LAST_NAME        FIRST_NAME
 Wilson           Oscar
1 row selected
```

The RDMS$SET_FLAGS logical name or the SQL SET FLAGS statement can disable this default behavior by using the 'NOREWRITE(LIKE)' keywords. The default is SET FLAGS 'REWRITE(LIKE)'.

5.1.5 Additional Database Storage Area Checks

In rare cases generally involving concealed logical names or in a cluster environment, it was possible to construct cases where two physical databases could access the same storage area file in an uncoordinated fashion. This access could lead to data corruption.

The steps required for this type of uncoordinated access would include a database restore or copy likely in a cluster environment. Because the databases shared the same original creation time stamp, so too would the storage area. Oracle Rdb would not detect that the storage areas were for physically separate databases.

This problem has been corrected in Oracle Rdb Release 7.2.5. Oracle Rdb now stores a time stamp of the physical database creation (via copy, restore or create) in the database root file and each storage area file. Each new access to a storage area compares the time stamp of the database root file and the storage area file to further ensure that they are part of the same physical database. If a mismatch is detected, the message INVDBSFIL "inconsistent storage area file" is signaled.

5.1.6 New Optimizations for the STARTING WITH Predicate

This release of Oracle Rdb will try to rewrite the STARTING WITH predicate when the STARTING WITH string is a literal. This enhancement allows Rdb to simplify some STARTING WITH expressions resulting in better index use and usually reducing I/O and CPU time requirements for these queries.

- When the STARTING WITH string is a zero length literal value then it will cause the STARTING WITH to match all non−NULL values. This can cause an unnecessary favoring of an index lookup which consumes CPU time with little advantage to the application.
- When the STARTING WITH string literal is the exact length of the source expression then complete non−NULL values are selected and Rdb can replace the STARTING WITH with an equality (=) predicate.

```
SQL> select last_name, first_name
cont> from employees
cont> where last_name starting with 'Smith         ';
Tables:
  0 = EMPLOYEES
Leaf#01 FFirst 0:EMPLOYEES Card=100
  Bool: 0.LAST_NAME = 'Smith         '
  BgrNdx1 EMPLOYEES_LAST_NAME [1:1] Fan=12
    Keys: 0.LAST_NAME = 'Smith         '
 LAST_NAME        FIRST_NAME
 Smith            Terry
 Smith            Roger
2 rows selected
```

The RDMS$SET_FLAGS logical name or the SQL SET FLAGS statement can disable this default behavior by using the 'NOREWRITE(STARTING_WITH)' keywords. The default is SET FLAGS 'REWRITE(STARTING_WITH)';

5.1.7 New Optimizations for the CONTAINING Predicate

This release of Oracle Rdb will try to rewrite the CONTAINING predicate when the CONTAINING string is a literal value. This enhancement allows Rdb to simplify some CONTAINING expressions resulting in better

index use and usually reducing I/O and CPU time requirements for these queries.

- When the CONTAINING string is a zero length literal value then it will cause the CONTAINING to match all non−NULL values. This can consume CPU time with little advantage to the application. This is replaced by a semantically equivalent OCTET_LENGTH function reference.

The RDMS$SET_FLAGS logical name or the SQL SET FLAGS statement can disable this default behavior by using the 'NOREWRITE(CONTAINING)' keywords. The default is SET FLAGS 'REWRITE(CONTAINING)'.

5.1.8 Monitor Memory Management Enhancements

Previously, the Oracle Rdb Monitor (RDMMON) process would map each database global (TROOT) section into P0 virtual address space. This could, in some cases, consume a significant portion of the 1GB available space and could also result in the virtual address space becoming sufficiently fragmented such that the monitor would be unable to open a database.

As a possible workaround, the monitor process can be restarted.

The impact of this virtual memory fragmentation has been somewhat reduced. The Oracle Rdb Monitor (RDMMON) process now maps database global sections (those that use SHARED MEMORY IS PROCESS or SHARED MEMORY IS PROCESS RESIDENT) into 64−bit P2 virtual address space. In addition, on OpenVMS Integrity Server systems, the executable code of the Oracle Rdb Monitor (RDMMON) process is mapped into 64−bit P2 virtual address space further reducing the amount of P0 virtual address space consumed.

5.1.9 Average Transaction Duration Display Precision Increased

As systems and applications become faster, it is more common for transaction durations of less than .0000 seconds to be performed. Currently, it is not possible to accurately determine, with the RMU /SHOW STATISTICS utility, the average duration of such rapid transactions. This condition has been addressed.

The average transaction duration value on the "Transaction Duration" display of the RMU /SHOW STATISTICS utility has been increased in precision from 4 to 6 fractional digits and the output display slightly modified to fit into an 80 column display, as in the following example.

```
Node: RDBTUK (1/1/1) Oracle Rdb V7.2−50 Perf. Monitor 29−SEP−2010 23:10:29.65
Rate: 3.00 Seconds     Transaction Duration (Total)      Elapsed: 00:00:26.77
Page: 1 of 1          DISK$TUKWILA7:[DB.V72]FOO.RDB;1            Mode: Online
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Total transaction count:            771
Seconds Tx.Count:   %  #Complete:   % #Incomplete:   %
 0−< 1:       773 100%         773 100%            0   0% <−avg=0.000192 95%=0.0
 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%
```

5.1.10 Support for New CONCAT_WS Builtin Function

This release of Oracle Rdb adds support for a new builtin function, CONCAT_WS, which is a variation of the CONCAT function. This function uses the first parameter as a separator which is applied after each of the other parameters. For instance, to create a comma separated list of column values, you specify the separator once and have SQL perform the formatting. This is shown in the example below.

```
SQL> select CONCAT_WS (', ', employee_id, birthday, '',
cont>                   last_name, middle_initial, last_name)
cont> from employees
cont> order by employee_id
cont> fetch first 10 rows only;

 00164, 1947-03-28, , Toliver       , A, Toliver
 00165, 1954-05-15, , Smith         , D, Smith
 00166, 1954-03-20, , Dietrich      , Dietrich
 00167, 1937-03-05, , Kilpatrick    , Kilpatrick
 00168, 1932-10-23, , Nash          , Nash
 00169, 1938-08-13, , Gray          , O, Gray
 00170, 1957-06-03, , Wood          , Wood
 00171, 1932-01-29, , D'Amico       , D'Amico
 00172, 1951-05-31, , Peters        , K, Peters
 00173, 1927-03-05, , Bartlett      , G, Bartlett
10 rows selected
SQL>
```

Usage Notes

- If the separator value expression resolves to NULL, then the result of CONCAT_WS will be NULL.
- If any other parameter value expression resolves to NULL, then it will be ignored. That is, that column value and any separator will not be included in the output.
- The function CONCAT_WS accepts all data types with the exception of LIST OF BYTE VARYING, LONG, and LONG RAW. Each non-character string value will be implicitly converted to VARCHAR with a size appropriate for the data type.
  The result of this function will have the type VARCHAR with a length long enough for the concatenated data and separators.
- If dialect ORACLE LEVEL1 or ORACLE LEVEL2 is used, then zero length strings ('') will be considered as NULL and so be excluded from the output. If the resulting value is a zero length string, then the result of CONCAT_WS will be NULL.

Examples

This example shows the use of the CONCAT_WS function to simplify the formatting of table data in CSV (comma separated value) format.

```
SQL> select '"' ||
cont> CONCAT_WS ('", "', first_name, nvl(middle_initial,''), last_name)
cont> || '"'
cont> from employees
cont> order by employee_id;
```

```
 "Alvin     ", "A", "Toliver       "
 "Terry     ", "D", "Smith         "
 "Rick      ", "", "Dietrich      "
 "Janet     ", "", "Kilpatrick    "
...
 "Peter     ", "", "Blount        "
 "Johanna   ", "P", "MacDonald     "
 "James     ", "Q", "Herbener      "
100 rows selected
SQL>
```

5.1.11 New SYSTIMESTAMP Function Added

This release of Oracle Rdb includes a new SYSTIMESTAMP function that returns the current date and time as a TIMESTAMP type. This function is similar to SYSDATE and CURRENT_TIMESTAMP however its type doesn't change when the SET DEFAULT DATE FORMAT command is used.

Syntax

SYSTIMESTAMP [(fractional−seconds−precision)]

Usage Notes

- The function name can be followed by an optional fractional−seconds−precision. This value, if omitted, defaults to 2 and accepts the values 0, 1, or 2.
- The following example shows that SYSTIMESTAMP always returns a SQL standard date and time.

```
SQL> select systimestamp,sysdate,current_timestamp from rdb$database;

 2007−03−27 16:33:32.19   27−MAR−2007 16:33:32.19   27−MAR−2007 16:33:32.19
1 row selected
SQL> set default date format 'sql99';
SQL> select systimestamp,sysdate,current_timestamp from rdb$database;

 2007−03−27 16:33:41.32   2007−03−27 16:33:41.32   2007−03−27 16:33:41.32
1 row selected
SQL>
```

5.1.12 New SET FLAGS Keyword to Control Optimizer Query Rewrite

This release of Oracle Rdb introduces several query rewrite optimizations. These optimizations are enabled by default and can be controlled using the RDMS$SET_FLAGS logical name or the SET FLAGS statement.

The following keywords or keyword parameters can be used. Note that REWRITE, unlike many other SET FLAGS keywords, does not accept a numeric value.

- REWRITE − when no parameters are provided, all query rewrite optimizations are enabled.
- NOREWRITE − when no parameters are provided, all query rewrite optimizations are disabled.
- REWRITE(LIKE), NOREWRITE(LIKE) − specifying the LIKE keyword will enable or disable only the LIKE predicate rewrite.

- REWRITE(STARTING_WITH), NOREWRITE(STARTING_WITH) − specifying the STARTING_WITH keyword will enable or disable only the STARTING WITH predicate rewrite.
- REWRITE(CONTAINING), NOREWRITE(CONTAINING) − specifying the CONTAINING keyword will enable or disable only the CONTAINING predicate rewrite.
- When SET FLAGS 'NONE' or SET NOFLAGS is used, the default setting for REWRITE will be restored.

The following example uses SET and SHOW FLAGS to show the effect of using the NOREWRITE keyword.

```
SQL> set line length 70
SQL> show flags;

Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION,MAX_RECURSION(100)
   ,REWRITE(CONTAINING),REWRITE(LIKE),REWRITE(STARTING_WITH)
   ,REFINE_ESTIMATES(127),NOBITMAPPED_SCAN
SQL>
SQL> set flags 'norewrite';
SQL> show flags;

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

5.1.13 New SYS_GUID Function Added

This release of Oracle Rdb supports a new builtin function, SYS_GUID. This function returns a 16 octet globally unique identifier. Applications would use this to provide unique values from various applications and across databases in an OpenVMS cluster or network.

Syntax

SYS_GUID ( )

Usage Notes

- This function uses the OpenVMS system service SYS$CREATE_UID. Applications that call this system service create compatible values for Rdb.
- The returned value from SYS_GUID() may contain octets that are zero. If returning values to C applications, then Oracle recommends using the $SQL_VARCHAR pseudo−type to avoid C null terminated string semantics.
- The SYS_GUID() returns data using a special character set. This special character set is used by Oracle Rdb to distinguish this type of string from others. Interactive SQL will format the value using standard OpenVMS formatting services when this character set is seen. Note that these services perform reordering of the octet values during formatting. That is, the value is not a direct hexadecimal representation of the value.
  Database administrators can define a domain to be used by applications which will make it easier to use.

  ```
  SQL> create domain GUID_DOMAIN
  ```

```
cont>    char(16) character set −11;
SQL>
SQL show domain GUID_DOMAIN;
GUID_DOMAIN                     CHAR(16)
         GUID 16 Characters,  16 Octets
SQL>
```

This domain can be used for column, parameter, and variable definitions.

- To support storing literal GUID values, SQL also supports GUID literals. The literals follow the standard literal format using the special prefix _GUID, as shown in the following examples.

```
SQL> create domain GUID_DOMAIN
cont>    char(16) character set −11;
SQL> show domain GUID_DOMAIN;
GUID_DOMAIN                     CHAR(16)
         GUID 16 Characters,  16 Octets

SQL> create table SAMPLE
cont>     (a int
cont>     ,b GUID_DOMAIN default _guid'00000000−0000−0000−0000−000000000000');
SQL> insert into SAMPLE default values;
1 row inserted

SQL> show table (column) SAMPLE;
Information for table SAMPLE

Columns for table SAMPLE:
Column Name                     Data Type        Domain
−−−−−−−−−−−                     −−−−−−−−−        −−−−−−
A                               INTEGER
B                               CHAR(16)         GUID_DOMAIN
         GUID 16 Characters,  16 Octets
 Oracle Rdb default: GUID'00000000−0000−0000−0000−000000000000'
SQL>
```

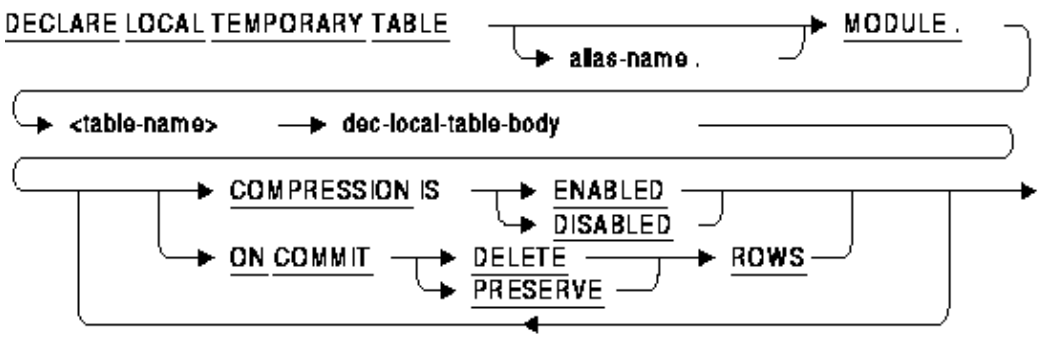
The literal can also be used in queries to select existing rows.

```
SQL> select * from SAMPLE
cont> where b = _guid'3DBB657F−8513−11DF−9B74−0008029189E7';
```

5.1.14 New COMPRESSION Clause for DECLARE LOCAL TEMPORARY TABLE Statement

This release of Oracle Rdb enhances the DECLARE LOCAL TEMPORARY TABLE syntax with a new COMPRESSION option. In prior releases of Rdb, it was not possible to disable row compression for temporary tables even though doing so might save some compression and decompression CPU time overhead. In addition, if the data in the temporary table is not compressible, it is possible that the run−length encoding could increase the resulting row length.

Syntax



Parameters

- COMPRESSION IS ENABLED
  COMPRESSION IS DISABLED
  This clause controls the use of run−length compression for rows inserted into this local temporary table. The default is COMPRESSION IS ENABLED.

Usage Notes

- In some cases, the data inserted into a local temporary table may not compress and so incur only overhead in the row. This overhead is used by Rdb to describe the sequence of uncompressible data. Use COMPRESSION IS DISABLED to prevent Rdb from attempting the compression of such data.

Examples

The following example shows a declared local temporary table that will not benefit from compression. The clause COMPRESSION IS DISABLED is used to reduce the CPU overhead for the table as well as preventing a possible row size increase because of compression notations.

```
SQL> declare local temporary table module.scratch0
cont>     (averages double precision)
cont>     compression is DISABLED
cont>     on commit PRESERVE rows
cont> ;
SQL>
SQL> insert into module.scratch0
cont>     select avg (char_length (a)) from module.scratch1;
1 row inserted
SQL>
SQL> select * from module.scratch0;
                AVERAGES
  2.100000000000000E+001
```

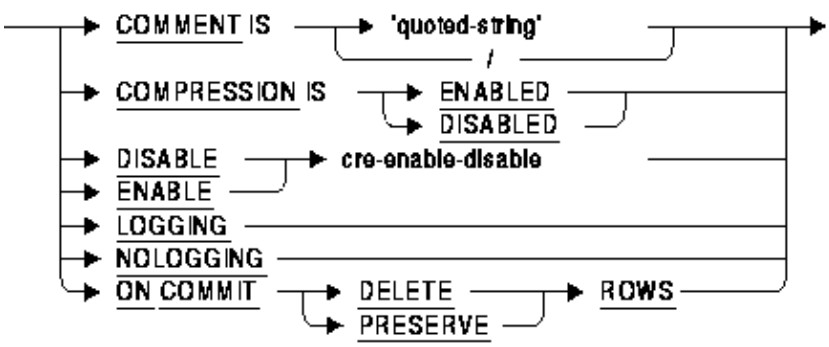
5.1.15 New COMPRESSION Clause for CREATE TABLE Statement

In prior releases of Rdb, it was required that a storage map be created for the table so that row compression

could be disabled. This release of Oracle Rdb enhances the CREATE TABLE syntax with a new COMPRESSION option.

Syntax

create-table-attributes =



Parameters

- COMPRESSION IS ENABLED
  COMPRESSION IS DISABLED
  This clause controls the use of run−length compression for rows inserted into this base or temporary table. The default is COMPRESSION IS ENABLED.

Usage Notes

- In some cases, the data inserted into a table may not compress and so incur only overhead in the row. This overhead is used by Rdb to describe the sequence of uncompressible data. Use COMPRESSION IS DISABLED to prevent Rdb from attempting the compression of such data.
- Any storage map which specifies the ENABLE COMPRESSION or DISABLE COMPRESSION clause will override this setting in the table.
- The COMPRESSION IS clause is not permitted for INFORMATION tables.

Examples

The following example shows that compression was disabled for the created table. The SHOW TABLE statement reports the disabled (that is the non−default) setting for compression.

```
SQL> create table SAMPLE
cont> (ident integer identity
cont> ,sample_value real
cont> )
cont> compression is disabled;
SQL> show table SAMPLE
Information for table SAMPLE

Compression is disabled.
Columns for table SAMPLE:
Column Name                    Data Type        Domain
```

```
-----------                     ---------       ------
IDENT                           INTEGER
 Computed:      IDENTITY
SAMPLE_VALUE                    REAL

Table constraints for SAMPLE:
 No Constraints found

Constraints referencing table SAMPLE:
 No Constraints found

Indexes on table SAMPLE:
 No Indexes found

Storage Map for table SAMPLE:
 No Storage Map found

Triggers on table SAMPLE:
No triggers found

SQL>
```

5.1.16 Support for 2 TiB Storage Area Files

OpenVMS Version 8.4 provides support for disk volumes up to 2 TiB in size. The precise maximum volume size is 4,261,348,350 blocks, which is about 1.98 TiB. Oracle Rdb now supports live and snapshot storage areas with up to 2,147,483,647 database pages and up to the OpenVMS Version 8.4 file size limit of 4,261,348,350 blocks. In order to utilize a database storage area of more than 2,147,483,647 disk blocks (approximately 1 TiB), a database page size of greater than 1 is required; in all cases, a database storage area is limited to 2,147,483,647 pages.

For Oracle Rdb Release 7.2.5, 2 TiB file support is limited to live and snapshot storage areas. Future Oracle Rdb releases are expected to expand this support to cover additional on−disk component files.

Note

The tebibyte is a standards−based binary multiple (prefix tebi, symbol Ti) of the byte, a unit of digital information storage. The tebibyte unit symbol is TiB.

```
1 tebibyte = 2^40 bytes = 1099511627776 bytes = 1024 gibibytes
```

The tebibyte is closely related to the terabyte, which is defined as 10^12 bytes or 1000000000000 bytes, but has been used as a synonym for tebibyte in some contexts.

5.1.17 New RMU/ALTER Feature to Modify the Root and Area Header Unique Identifier

To ensure Oracle Rdb database security and integrity, a Unique Identifier has been added to the database root file and the database storage area file and storage area snapshot file headers. The Unique Identifier in the root

file must match the Unique Identifier in the storage area file headers or a storage area cannot be accessed from the database root. The RMU/ALTER command has been enhanced to allow a Database Administrator to modify and display the database root and storage area Unique Identifier values to prevent problems which will occur if these values are corrupted.

The Unique Identifier values are displayed both in VMS date format surrounded by quotes and as a hexadecimal number surrounded by parentheses. The values displayed are the Unique Identifier values for the current RMU/ALTER session. The Unique Identifier values will not be written to the root or storage area files until the user ends the current session with the RMU/ALTER "COMMIT" command. If the user ends the current session with the RMU/ALTER "ROLLBACK" command, the Unique Identifier values will not be written to the root or storage area files and the Unique Identifier values in effect at the start of the session just ended will be restored for the new session. Any Unique Identifier values that have been changed during the current session will be displayed as "(marked)" before they are committed or rolled back.

The new syntax, which can only be used at the "RdbALTER>" prompt which appears when the RMU/ALTER command is issued at the VMS prompt, is the following where "name" is the storage area name and "id" is the storage area identification number in the database root file. "AREA_HEADER" refers to the storage area file header. "ROOT" refers to the Rdb database root file (*.RDB). "UNIQUE_IDENTIFIER" refers to the Unique Identifier in the storage area header when used with "AREA_HEADER" and to the Unique Identifier in the database root when used with "ROOT". If "SNAPSHOT" is not specified, the storage area (*.RDA) file is assumed. If "SNAPSHOT" is specified, the snapshot storage area (*.SNP) file is assumed. The user cannot specify a new "UNIQUE_IDENTIFIER" value: it must be created by RMU/ALTER.

```
DISPLAY ROOT UNIQUE_IDENTIFIER
```

This command displays the current database root Unique Identifier value.

```
DEPOSIT ROOT UNIQUE_IDENTIFIER (= NEW)
```

If "= NEW" is not specified, this command stores the current database root Unique Identifier value into the storage area header blocks of ALL active storage area and storage area snapshot files which are currently defined in the database root when the user executes the next COMMIT command. If "= NEW" is specified, a new Unique Identifier value is created and stored in both the root file and ALL active storage area file headers when the user executes the next COMMIT command. Note that to ensure database integrity, ALL storage area file headers will be updated. Use the AREA_HEADER commands described below for storing the current root Unique Identifier value in specific designated storage areas.

```
DISPLAY AREA_HEADER {name|id} (SNAPSHOT) UNIQUE_IDENTIFIER
```

This command displays the current storage area file or storage area snapshot file header Unique Identifier value for the storage area with the specified name or number.

```
DEPOSIT AREA_HEADER {name|id} (SNAPSHOT) UNIQUE_IDENTIFIER
```

This command stores the current database root Unique Identifier value into the current storage area file or storage area snapshot file header for the storage area with the specified name or number when the user executes the next COMMIT command.

As stated above, any changes to the area header or root Unique Identifier values will only be written to the actual root and area files when the next "COMMIT" command is executed at the "RdbALTER>" prompt. Any changes to the root or 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 uncomitted changes made in RMU/ALTER, not just changes to the root and storage area header Unique Identifier values.

To execute the DISPLAY or DEPOSIT ROOT or AREA_HEADER command, the user must be attached to the database which the root and 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 following example shows that for Oracle Rdb single file databases, the Unique Identifier value can only be set for the storage area snapshot file since the storage area is part of the root file. Therefore, the DEPOSIT AREA_HEADER command can only specify the "SNAPSHOT" file or an error will be returned.

```
$ RMU/ALTER PERSONNEL
%RMU-I-ATTACH, now altering database
"DEVICE:[DIRECTORY]PERSONNEL.RDB;1"
  DEPOSIT AREA_HEADER RDB$SYSTEM UNIQUE_IDENTIFIER
%RMU-F-NOTSFDB, This command is not allowed for a single file
 database
  DEPOSIT AREA_HEADER RDB$SYSTEM SNAPSHOT UNIQUE_IDENTIFIER
Area RDB$SYSTEM:
(marked) Root file unique identifier is: "22-OCT-2010 13:49:29.32"
 (00AA557869612643)

 COMMIT
 EXIT
```

Since the DEPOSIT ROOT UNIQUE_IDENTIFIER command always stores the Unique Identifier value in ALL storage area file headers when the user executes the RMU/ALTER "COMMIT" command, it would be redundant to execute the DEPOSIT AREA_HEADER UNIQUE_IDENTIFIER command if a DEPOSIT ROOT UNIQUE_IDENTIFIER command is already pending for the current RMU/ALTER session. Therefore, as the following example shows, in this case a DEPOSIT AREA_HEADER UNIQUE_IDENTIFIER command cannot be executed until the user ends the current session with a COMMIT or ROLLBACK command.

```
$ RMU/ALTER MF_PERSONNEL
%RMU-I-ATTACH, now altering database
 "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1"
  DEPOSIT ROOT UNIQUE_IDENTIFIER = NEW
(marked) Root file unique identifier is: "22-OCT-2010 13:49:31.72"
 (00AA55786ACFB115)

  DEPOSIT AREA_HEADER SALARY_HISTORY UNIQUE_IDENTIFIER
%RMU-F-COMROOTCOM, COMMIT or ROLLBACK DEPOSIT ROOT
 UNIQUE_IDENTIFIER command to use this command
  commit
  DEPOSIT AREA_HEADER SALARY_HISTORY UNIQUE_IDENTIFIER
Area SALARY_HISTORY:
(marked) Root file unique identifier is: "22-OCT-2010 13:49:31.72"
 (00AA55786ACFB115)

  COMMIT
  EXIT
```

The following example shows that RMU/ALTER is invoked specifying the database MF_PERSONNEL.RDB. The user then displays the current Unique Identifier value in the database root, creates a new Unique Identifier value in the database root, displays the new Unique Identifier in the root, and

finally specifies "commit" to write the new Unique Identifier value to the database root file and ALL database storage area files. The display messages designate the pending new Unique Identifier value as "(marked)" until the user either executes "commit" to write out the new Unique Identifier value or "rollback" to restore the original Unique Identifier value. The user then verifies the database changes.

```
$ RMU/ALTER MF_PERSONNEL
%RMU-I-ATTACH, now altering database "DISK:[DIRECTORY]MF_PERSONNEL.RDB;1"
  DISPLAY ROOT UNIQUE_IDENTIFIER
         Root file unique identifier is: "22-OCT-2010 13:49:27.87"
 (00AA5578688428BB)
  DEPOSIT ROOT UNIQUE_IDENTIFIER = NEW
(marked) Root file unique identifier is: "22-OCT-2010 13:49:28.34"
 (00AA557868CC9F7A)
  DISPLAY ROOT UNIQUE_IDENTIFIER
(marked) Root file unique identifier is: "22-OCT-2010 13:49:28.34"
 (00AA557868CC9F7A)
  COMMIT
  EXIT
$ RMU/VERIFY/ALL/NOLOG MF_PERSONNEL
```

The following example shows that RMU/ALTER is invoked specifying the database MF_PERSONNEL.RDB. The user then displays the current Unique Identifier value in the root file. He then executes the "deposit" commands to designate that the Unique Identifer value in the root file is to be moved to the DEPARTMENTS area storage and the DEPARTMENTS area snapshot files, displays the Unique Identifier value that is to be moved to the DEPARTMENTS area storage and the DEPARTMENTS area snapshot files, and finally specifies "commit" to actually write the root unique identifier value to the DEPARTMENTS area storage and the DEPARTMENTS area snapshot files. The display messages designate the pending Unique Identifier value as "(marked)" until the user either executes "commit" to write out the Unique Identifier value or "rollback" to restore the original Unique Identifier value. The user then verifies the database changes. The example shows that the user can use either the storage area name or the storage area identifier number in the root to designate the target storage area.

```
$ RMU/ALTER MF_PESONNEL
%RMU-I-ATTACH, now altering database "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1"
  DISPLAY ROOT UNIQUE_IDENTIFIER
           Root file unique identifier is: "22-OCT-2010 13:49:28.34"
 (00AA557868CC9F7A)

  DEPOSIT AREA_HEADER DEPARTMENTS UNIQUE_IDENTIFIER
Area DEPARTMENTS:
(marked) Root file unique identifier is: "22-OCT-2010 13:49:28.34"
 (00AA557868CC9F7A)

  DEPOSIT AREA_HEADER 2 SNAPSHOT UNIQUE_IDENTIFIER
Area DEPARTMENTS:
(marked) Root file unique identifier is: "22-OCT-2010 13:49:28.34"
 (00AA557868CC9F7A)

  DISPLAY AREA_HEADER DEPARTMENTS UNIQUE_IDENTIFIER
Area DEPARTMENTS:
(marked) Root file unique identifier is: "22-OCT-2010 13:49:28.34"
 (00AA557868CC9F7A)

  DISPLAY AREA_HEADER 2 SNAPSHOT UNIQUE_IDENTIFIER
Area DEPARTMENTS:
(marked) Root file unique identifier is: "22-OCT-2010 13:49:28.34"
 (00AA557868CC9F7A)
```

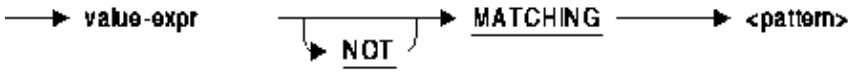
```
  COMMIT
  EXIT
$ RMU/VERIFY/ALL/NOLOG MF_PERSONNEL
```

5.1.18 New MATCHING Predicate

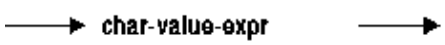
This release of Oracle Rdb supports MATCHING as an alternate string pattern matching clause.

Syntax

matching-predicate =



pattern =



A MATCHING predicate searches character string literals for pattern matches. The pattern string accepts the following pattern characters:

- * Matches any string of zero or more characters
- % Matches any single character

Usage Notes

- If either of the expressions is null, the result is null.
- The MATCHING predicate is not case sensitive; it considers uppercase and lowercase forms of the same character to be a match.
- The MATCHING predicate is not sensitive to diacritical markings used in the DEC Multinational Character Set.

The following example shows the use of the MATCHING clause.

```
SQL> select last_name
cont> from employees
cont> where last_name matching '%on*';
 LAST_NAME
 Connolly
 Lonergan
2 rows selected
SQL>
```

5.1.19 New RMU/BACKUP−RESTORE Feature to Check Database Page Integrity

To ensure Oracle Rdb database integrity, a new feature has been added to the RMU/BACKUP and RMU/RESTORE commands to do a basic integrity check of AIP, ABM and data storage area pages when they are backed up and when they are restored. SPAM pages are not backed up by RMU/BACKUP but recreated by RMU/RESTORE based on the page types that are backed up. This basic integrity check will always happen in order to prevent database corruption.

The basic page checks made are not intended to replace an RMU/VERIFY of the database pages before they are backed up or after they are restored. If the page checks report problems, the user should immediately do an RMU/VERIFY of the storage area to get a complete evaluation of the detected problems and any additional problems that may exist. The page checks made are intended to cause minimal additional overhead to the backup or restore operation while indicating serious page corruption exists that needs to be investigated using the RMU/VERIFY and/or RMU/DUMP commands and then corrected. The page checks are necessary so that page integrity problems can be detected and fixed at backup time or so that page integrity problems can be reported and fixed after the restore.

The checks made are a check for a valid storage area id number on the page, a check for a non−zero timestamp on the page, and a check for a page number that is greater than zero and less than or equal to the maximum page number for the storage area. The diagnostic messages output are the same as the diagnostic messages output by RMU/VERIFY if these problems are detected. For RMU/BACKUP, the backup is aborted if any of these checks fail so that the problem can be fixed at backup time. For RMU/RESTORE, the restore will be aborted only if the page number check fails since RMU/RESTORE cannot continue in this case. The other problems will be reported but the backup will continue so the problems can then be fixed in the restored database. Neither the backup or restore operation will be aborted until all the specified checks have been made.

The following example shows that the backup command is aborted because of a page with an invalid page number in storage area SALARY_HISTORY. An RMU/VERIFY of the database gives the additional information of the expected page number where the invalid page number of zero occurs.

```
$ RMU/BACKUP/NOLOG MF_PERSONNEL.RDB MFP.RBF
%RMU−E−BADPAGRAN, area SALARY_HISTORY
%RMU−E−BADPAGRA2,                   page number 0 out of range
%RMU−E−BADPAGRA3,                   expected between 1 and 706
%RMU−F−INVPAGAREA, Aborting command − invalid page 0 in area SALARY_HISTORY
%RMU−F−FATALERR, fatal error on BACKUP
%RMU−F−FTL_BCK, Fatal error for BACKUP operation at 21−DEC−2010
 09:30:19.64
$ rmu/verify/all/nolog mf_personnel
%RMU−W−PAGPAGRAN, area SALARY_HISTORY, page 21
                  page number out of range
                  expected: 21, found: 0
```

The following database backup example shows all three diagnostics that can be put out by the basic page validation checks. If any of these checks fail, the backup operation will be aborted.

```
$ RMU/BACKUP/NOLOG MF_PERSONNEL.RDB MFP.RBF
%RMU−W−PAGBADARE, area RDB$SYSTEM, page 0
%RMU−W−PAGBADAR2,                   maps incorrect storage area
%RMU−W−PAGBADAR3,                   expected: 1, found: 3
%RMU−W−PAGTADZER, area RDB$SYSTEM, page 0
```

```
%RMU−W−PAGTADZE2,                   contains zero time stamp
%RMU−E−BADPAGRAN, area RDB$SYSTEM
%RMU−E−BADPAGRA2,                   page number 0 out of range
%RMU−E−BADPAGRA3,                   expected between 1 and 1012
%RMU−F−INVPAGAREA, Aborting command − invalid page 0 in area RDB$SYSTEM
%RMU−F−FATALERR, fatal error on BACKUP
%RMU−F−FTL_BCK, Fatal error for BACKUP operation at 21−DEC−2010 09:39:21.64
```

The following database restore example shows all three diagnostics that can be put out by the basic page validation checks. The restore operation will be aborted only if the page number is invalid but the other reported problems must be corrected.

```
$ RMU/RESTORE/NOLOG/NOCDD MFP.RBF
%RMU−W−PAGBADARE, area RDB$SYSTEM, page 0
%RMU−W−PAGBADAR2,                   maps incorrect storage area
%RMU−W−PAGBADAR3,                   expected: 1, found: 3
%RMU−W−PAGTADZER, area RDB$SYSTEM, page 0
%RMU−W−PAGTADZE2,                   contains zero time stamp
%RMU−E−BADPAGRAN, area RDB$SYSTEM
%RMU−E−BADPAGRA2,                   page number 0 out of range
%RMU−E−BADPAGRA3,                   expected between 1 and 1012
RMU−F−INVPAGAREA, Aborting command − invalid page 0 in area RDB$SYSTEM
%RMU−F−FATALERR, fatal error on RESTORE
%RMU−F−FTL_RSTR, Fatal error for RESTORE operation at 21−DEC−2010 09:44:01.96
```

5.1.20 New RMU/DUMP/BACKUP /AREA, /START and /END Qualifiers

New /AREA, /START and /END qualifiers have been added to the Oracle Rdb RMU/DUMP/BACKUP command which dumps the contents of an Rdb database backup (*.RBF) file. These qualifiers allow the user to dump backup file records for a specified storage area and for a specified range of pages within the specified storage area.

The syntax for the new qualifiers used with the RMU/DUMP/BACKUP command is the following.

```
/AREA = identity
```

Only dump the storage area identified by the specified name or ID number. The area name must be the name of a storage area in the database root file and the area ID number must be a storage area ID number in the database root file. This information is contained in the "Database Parameters:" section of the backup file which is output at the start of the dump. Snapshot areas are not contained in the backup file and cannot be specified. If this qualifier is used without the new /START and /END qualifiers, all page records in the specified storage area will be output.

```
/START = number
```

Only dump pages starting with the specified page number in the specified storage area. This qualifier cannot be used unless the /AREA qualifier is also specified. If no pages are dumped either the specified page or range of pages does not exist in the specified area in the backup file, or this qualifier has been used in the same RMU/DUMP/BACKUP command as an /OPTIONS, /SKIP or /PROCESS qualifier option that has excluded the specified page or range of pages from the dump. If this qualifier is not used with the new /END qualifier, all page records in the specified storage area starting with the specified page number will be output.

```
/END = number
```

Only dump pages ending with the specified page number in the specified storage area. This qualifier cannot be used unless the /AREA qualifier is also specified. If no pages are dumped either the specified page or range of pages does not exist in the specified area in the backup file, or this qualifier has been used in the same RMU/DUMP/BACKUP command as an /OPTIONS, /SKIP or /PROCESS qualifier option that has excluded the specified page or range of pages from the dump. If this qualifier is not used with the new /START qualifier, all page records in the specified storage area ending with the specified page number will be output.

If both the /START and /END qualifiers are specified, the starting page number must be less than or equal to the ending page number. If the starting page number equals the ending page number only the page records for the specified page number are dumped. The block header for each block which contains at least one of the requested pages is dumped followed by the requested page records in that block. The START AREA record is dumped at the start of requested page records and the END AREA record is dumped at the end of the requested page records. By default, the database root parameters are dumped at the very start following the dump header.

The following example shows the dump of the page records for page 10 in storage area 4 in the MFP.RBF backup file. Since the /START and /END qualifiers both specify page 10, only the page records for that page are dumped. At the start of the dump is the dump header, followed by the database root parameters which are not shown to save space, followed by the block header, which begins with the "HEADER_SIZE" field, for the block which contains the records for page 10 in storage area 4, followed by the start area record for area 4 (REC_TYPE = 6), the data page header record (REC_TYPE = 7) for page 10, the data page data record (REC_TYPE = 8) for page 10, and ending with the end area record for area 4 (REC_TYPE = 11) which ends the dump.

```
$ RMU/DUMP/BACKUP/AREA=4/START=10/END=10/OPTION=FULL MFP.RBF
*-------------------------------------------------------------------------------
* Oracle Rdb V7.2-420                                    11-JAN-2011 15:50:09.25
*
* Dump of Database Backup Header
*     Backup filename: MFP.RBF
*     Backup file database version: 7.2
*
*-------------------------------------------------------------------------------


Database Parameters:
.
.
.

HEADER_SIZE = 80        OS_ID = 1024    UTILITY_ID = 722
APPLICATION_TYPE = 1    SEQUENCE_NUMBER = 22    MAJ_VER = 1    MIN_VER = 1
VOL_NUMBER = 1  BLOCK_SIZE = 32256      CRC = 0C5D3A78  NOCRC = 00
CRC_ALTERNATE = 00      BACKUP_NAME = MFP.RBF   AREA_ID = 4    HIGH_PNO = 259
LOW_PNO = 1     HDR_CHECKSUM = 9B3D

REC_SIZE = 2    REC_TYPE = 6    BADDATA = 00    ROOT = 00
AREA_ID = 4     LAREA_ID = 0    PNO = 0

REC_SIZE = 32   REC_TYPE = 7    BADDATA = 00    ROOT = 00
AREA_ID = 4     LAREA_ID = 0    PNO = 10

REC_SIZE = 28   REC_TYPE = 8    BADDATA = 00    ROOT = 00
AREA_ID = 4     LAREA_ID = 0    PNO = 10
```

```
REC_SIZE = 512  REC_TYPE = 11   BADDATA = 00    ROOT = 00
AREA_ID = 4     LAREA_ID = 0    PNO = 0
```

The following example dumps the records for pages 10, 11 and 12 in the RDB$SYSTEM storage area in the MFP.RBF backup file. Following the block header containing the target records that starts with "HEADER_SIZE =", are the start area record for RDB$SYSTEM area 1 (REC_TYPE = 6), then the target ABM page records for pages 10, 11, and 12 (REC_TYPE = 10), and finally the end area record for area RDB$SYSTEM area 1 (REC_TYPE = 11) which ends the dump.

```
$ RMU/DUMP/BACKUP/AREA=RDB$SYSTEM/START=10/END=12/OPTION=FULL MFP.RBF
*-------------------------------------------------------------------------------
* Oracle Rdb V7.2-420                                    14-JAN-2011 14:40:46.88
*
* Dump of Database Backup Header
*     Backup filename: MFP.RBF
*     Backup file database version: 7.2
*
*-------------------------------------------------------------------------------

Database Parameters:
.
.
.

HEADER_SIZE = 80        OS_ID = 1024    UTILITY_ID = 722
APPLICATION_TYPE = 1    SEQUENCE_NUMBER = 1     MAJ_VER = 1     MIN_VER = 1
VOL_NUMBER = 1  BLOCK_SIZE = 32256      CRC = 8329C24B  NOCRC = 00
CRC_ALTERNATE = 00      BACKUP_NAME = MFP.RBF   AREA_ID = 1     HIGH_PNO = 178
LOW_PNO = 1     HDR_CHECKSUM = 40DE

REC_SIZE = 2    REC_TYPE = 6    BADDATA = 00    ROOT = 00       AREA_ID = 1
LAREA_ID = 0    PNO = 0

REC_SIZE = 10   REC_TYPE = 10   BADDATA = 00    ROOT = 00       AREA_ID = 1
LAREA_ID = 3    PNO = 10

REC_SIZE = 10   REC_TYPE = 10   BADDATA = 00    ROOT = 00       AREA_ID = 1
LAREA_ID = 4    PNO = 11

REC_SIZE = 10   REC_TYPE = 10   BADDATA = 00    ROOT = 00       AREA_ID = 1
LAREA_ID = 4    PNO = 12

REC_SIZE = 512  REC_TYPE = 11   BADDATA = 00    ROOT = 00       AREA_ID = 1
LAREA_ID = 0    PNO = 0
```

5.1.21 Reduced CPU Usage and Improved Performance

Several performance enhancements have been implemented in this release of Oracle Rdb. Most of these changes are either specific to applications running on I64 systems or will have a greater effect on I64 systems. These enhancements include improved code sequences for:

- Integer and floating point arithmetic operations
- Floating point comparison operations
- Floating point conversion operations

5.1.22 New Logical Name to Control Sizing of LIST OF BYTE VARYING Pointer Segments

This release of Oracle Rdb supports a new logical name, RDMS$BIND_SEGMENTED_STRING_PSEG_SIZING, that can be used to control the upper limit for the size of the pointer segment which is stored for LIST OF BYTE VARYING data.

In prior releases, the upper limit for a pointer segment was the free space on a page. If only a small number of LIST segments were inserted, then the pointer segment was trimmed to that required by the data for that column. However, when many relatively small segments were stored there would be one or more large segments stored that required Rdb to search for free space (essentially a free storage area page). The result was a high number of discarded pages − pages read from disk that contained free space but insufficient free space for the stored pointer segments. The solution is to define THRESHOLD values for the MIXED format storage area, or the LIST storage map for UNIFORM format storage areas so that Rdb has guidance on acceptable pages.

To simplify the management of LIST OF BYTE VARYING data, Rdb now supports this new logical to control INSERT behavior for pointer segments. The logical name RDMS$BIND_SEGMENTED_STRING_PSEG_SIZING can be defined to one of the following numeric values:

- 0
  Use the current page size as the upper limit. This remains the default behavior (as in previous releases) if this logical name is not defined.
- 1
  Use the maximum segment length for the current LIST OF BYTE VARYING column value. For instance, if the application always stored 100 octet values in the LIST then this will be used (plus record overhead) to limit the size of the pointer segments.
- 2
  Use the average segment length for the current LIST OF BYTE VARYING column value. This is a useful setting when segments are of different sizes, such as lines from a text document.

Usage Notes

- If any other value is used for the logical name, then it will be ignored and the setting will default to standard behavior.
- This logical name affects all tables including system tables.
- The effect may be more smaller pointer segments. This will translate to increased I/O counts. However, the benefit should be better placement in the storage area and elimination (or reduction) in discarded pages.
- If the average or maximum length of the data segments is too small, then Rdb will ensure that the pointer segments can store at least three pointers. When stored, the pointer segments will be trimmed to contain only valid data pointers.
- The stored data is compatible with all releases of Oracle Rdb, and the logical can be deassigned or redefined at any time.
- If the logical name RDMS$USE_OLD_SEGMENTED_STRING is defined as "T", "t", or "1" then Rdb will revert to chained style segmented strings. In that case, the value specified by RDMS$BIND_SEGMENTED_STRING_PSEG_SIZING will not be used.

5.1.23 RMU /BACKUP Performance Improvements

RMU /BACKUP performance has been improved by streamlining code sequences and reducing redundant data copies in memory. In some cases, reductions of up to 10% of CPU time have been realized.

5.1.24 New RMU/BACKUP/ENCRYPT "%RMU−I−ENCRYPTUSED" Message Added

The Oracle Rdb RMU/BACKUP/ENCRYPT and RMU/BACKUP/AFTER_JOURNAL/ENCRYPT commands create encrypted Rdb database backup files and Rdb database After Image Journal backup files using an encryption key. The same encryption key must be specified when the database is restored or recovered by the RMU/RESTORE/ENCRYPT and RMU/RECOVER/ENCRYPT commands. The following informational message will now always be output on a successful completion of the RMU/BACKUP/ENCRYPT and RMU/BACKUP/AFTER_JOURNAL/ENCRYPT commands to remind the user that the same key specified for these commands must be specified when the created backup files are restored or recovered by the RMU/RESTORE/ENCRYPT and RMU/RECOVER/ENCRYPT commands.

```
%RMU−W−ENCRYPTUSED, Encryption key required when future restore
performed.
```

The following example shows this new informational message being put out when successful RMU/BACKUP/ENCRYPT and RMU/BACKUP/AFTER_JOURNAL/ENCRYPT commands are completed.

```
$ RMU/BACKUP/NOLOG database_file backup_file −
 /ENCRYPT=(VALUE=key_name,ALGORITHM=algorithm_name)
%RMU−I−ENCRYPTUSED, Encryption key required when future restore
performed.
$ RMU/BACKUP/AFTER_JOURNAL/FORMAT=NEW_TAPE −
 database_file backup_file −
 /ENCRYPT=(VALUE=key_name,ALGORITHM=algorithm_name)/NOLOG
%RMU−I−ENCRYPTUSED, Encryption key required when future restore
performed.
```

5.1.25 New DATABASE_HANDLE Option for the GET DIAGNOSTICS Statement

This release of Oracle Rdb adds the keyword DATABASE_HANDLE for use by the GET DIAGNOSTICS statement. This option returns a unique handle (or stream) identifier which can be useful in distinguishing one attach from another.

```
SQL> set flags 'trace';
SQL>
SQL> begin
cont> declare :ii integer;
cont> get diagnostics :ii = DATABASE_HANDLE;
cont> trace :ii;
cont> end;
~Xt: 1
SQL>
```

5.1.26 New SYS_GET_DIAGNOSTIC Function Supported for SQL

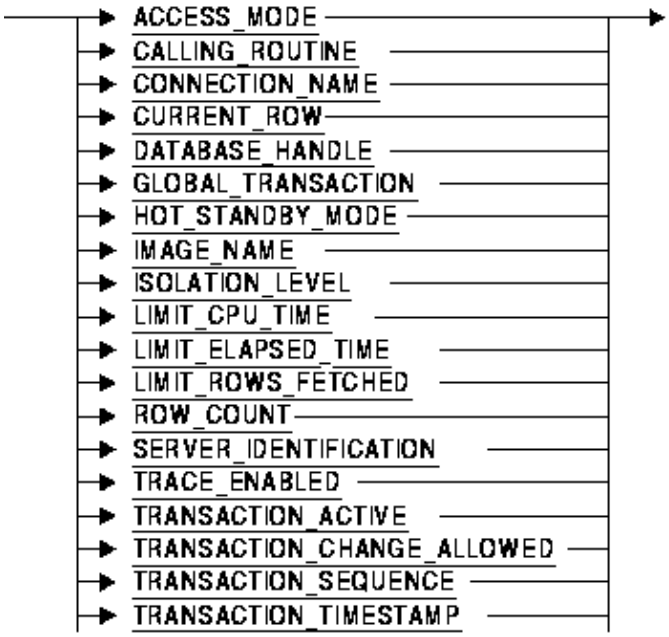
This release of Oracle Rdb adds a new function, SYS_GET_DIAGNOSTIC, that can be used to return the same session information available to the GET DIAGNOSTICS statement. This function provides a shorthand method of fetching values without requiring a compound statement or an intermediate variable.

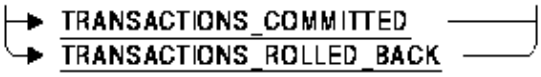
Syntax

SYS_GET_DIAGNOSTIC (statement−item−name)

---

statement-item-name =





Usage Notes

- For the list of keywords acceptable by this function, please see the statement−item−name syntax under the GET DIAGNOSTICS statement.
- Each keyword used with SYS_GET_DIAGNOSTIC will cause the result to have a different associated data type. Please refer to the GET DIAGNOSTICS statement for the returned data types.

Examples

The following example shows the return of session information.

```
SQL> select SYS_GET_DIAGNOSTIC (CONNECTION_NAME) as CONN,
cont>       SYS_GET_DIAGNOSTIC (SERVER_IDENTIFICATION) as IDENT,
cont>       SYS_GET_DIAGNOSTIC (DATABASE_HANDLE) as DBHANDLE
cont>  from GET_DIAG;
 CONN                              IDENT                               DBHANDLE
 RDB$DEFAULT_CONNECTION            Oracle Rdb V7.2−501                    1
1 row selected
SQL>
```

5.1.27 Improved Error Handling for Database Disk Backup File Sets

Bug 4596098

The "/DISK_FILE" command qualifier can be used with certain Oracle Rdb RMU commands such as "RMU/BACKUP" and "RMU/RESTORE" to create or read database disk backup file sets. The error handling for the "/DISK_FILE" qualifier used with the "RMU/RESTORE" and "RMU/DUMP/BACKUP" commands has been improved in the following cases.

If the first file of a backup file set is not specified, the "RMU/RESTORE" or "RMU/DUMP/BACKUP" command cannot continue since the first file of the backup file set contains the backed up root. For this case, a new message has been added which displays the backup command that was used to create the backup file set which is contained in the summary record at the start of each backup file. The backup command will show all the backup files created by the backup in the correct order.

```
%RMU-W-BCKCMDUSED, The backup command was
"RMU/BACKUP...".
```

In addition, a new message has been added to specify that file number "n" in the backup file set, the first backup set file specified in the restore command, is not the first backup file of the backup set created by the backup command.

```
%RMU-E-NOTFIRVOL, Backup set file number n, the first backup file specified,
is not the first file of the backup set. Specify all the backup set files or
devices in the correct order.
```

The following example shows the new error messages for this case. The first backup set file created by the "RMU/BACKUP/DISK_FILE" command, "MFP", is not specified by the "RMU/RESTORE" command.

```
$ RMU/RESTORE/NOLOG/NOCDD/DISK=READER=3 MFP01,MFP02,MFP03
%RMU-W-BCKCMDUSED, The backup command was
"RMU/BACKUP/NOLOG/DISK=WRITER=4 MF_PERSONNEL MFP,MFP01,MFP02,MFP03".
%RMU-E-NOTFIRVOL, Backup set file number 2, the first backup file specified,
is not the first file of the backup set. Specify all the backup set files or
devices in the correct order.
%RMU-F-INVDBBFIL, invalid backup file DEVICE:[DIRECTORY]MFP01.RBF;
%RMU-F-FATALERR, fatal error on RESTORE
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 22-MAR-2011 10:04:26.82
```

If the first file of the backup file set is specified but other file(s) of the set are left out, a new message has been added which displays the backup command that was used to create the backup file set which is contained in the summary record at the start of each backup file. This will show all the backup files created by the backup command in the correct order.

```
%RMU-W-BCKCMDUSED, The backup command was
"RMU/BACKUP...".
```

Also, the warning message below has been changed to the following error message.

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

```
%RMU-E-RESINCOMP, Not all storage areas have been restored, the
 database may be corrupt.
```

In this case, "%RMU−E−RESINCOMP" has been made the exit status. The restored database can be attached to and the storage areas that were restored can be accessed in SQL without error, though SQL will return an

error if access is attempted to the storage areas in the root that were not restored. The user should delete the restored files and repeat the restore using all the backup files in the backup set displayed in the "%RMU−W−BCKCMDUSED" message. If any of the backup files in the set have been lost, the missing storage areas can be restored by the RMU/RESTORE/AREA command using a previous backup file if it is available.

The new error messages for this case are not put out for the "RMU/DUMP/BACKUP" command since this command can be used to dump one or more files of a backup set without error as long as the first backup set file containing the root information is included as the first file or only file specified. If the "/OPTION=DEBUG" qualifier is used with the "RMU/DUMP/BACKUP" command, the backup command will be displayed along with the other summary record fields at the start of each backup set file.

The following example shows the new error messages for this case. The second backup set file created by the "RMU/BACKUP/DISK_FILE" command, "MFP01", is not specified by the "RMU/RESTORE" command.

```
$ RMU/RESTORE/NOLOG/NOCDD/DISK=READER=3 MFP,MFP02,MFP03
%RMU−I−AIJRSTAVL, 0 after−image journals available for use
%RMU−I−AIJISOFF, after−image journaling has been disabled
%RMU−W−USERECCOM, Use the RMU Recover command. The journals are not available.
%RMU−W−BCKCMDUSED, The backup command was
"RMU/BACKUP/NOLOG/DISK=WRITER=4 MF_PERSONNEL MFP,MFP01,MFP02,MFP03".
%RMU−E−RESINCOMP, Not all storage areas have been restored, the database may be
corrupt.
```

If, for the command "RMU/RESTORE/DISK_FILE/READER_THREADS=number", the number of reader threads specified was greater than the number of backup files specified, the following fatal message was put out and the restore was terminated. This message made sense only for tape backups where you can specify the "/VOLUMES" and "/MASTER" qualifiers.

```
%RMU−F−CONFLSWIT, conflicting qualifiers /VOLUMES and /MASTER
```

The %RMU−F−CONFLSWIT message has been replaced by the more general fatal error message below.

```
%RMU−F−EXTRAREADERS, "n", the number of reader threads, exceeds "n",
 the number of output files or master tape devices.
```

This message will not be displayed for the "RMU/DUMP/BACKUP" command which does not allow the number of reader threads to be specified.

The following example shows this last case. Five reader threads are specified for the "RMU/RESTORE/DISK_FILE" command but there are only four backup files in the backup file set so the first restore command fails. If four or fewer reader threads are specified, the command will succeed.

```
$ RMU/RESTORE/NOLOG/NOCDD/DISK=READER=5 MFP,MFP01,MFP02,MFP03
%RMU−F−EXTRAREADERS, "5", the number of reader threads, exceeds "4", the number
of output files or master tape devices.
%RMU−F−FTL_RSTR, Fatal error for RESTORE operation at 22−MAR−2011 10:17:08.23
$ RMU/RESTORE/NOLOG/NOCDD/DISK=READER=4 MFP,MFP01,MFP02,MFP03
%RMU−I−AIJRSTAVL, 0 after−image journals available for use
%RMU−I−AIJISOFF, after−image journaling has been disabled
%RMU−W−USERECCOM, Use the RMU Recover command. The journals are not available.
$ RMU/VERIFY/ALL MF_PERSONNEL
```

Chapter 6
Enhancements And Changes Provided in Oracle Rdb Release 7.2.4.2

6.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.4.2

6.1.1 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.

Chapter 7
Enhancements And Changes Provided in Oracle Rdb Release 7.2.4.1

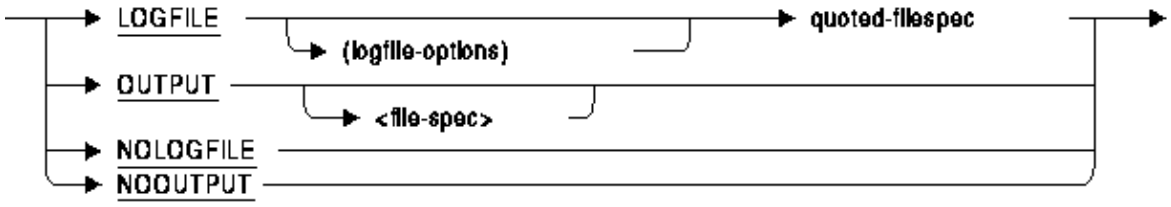
7.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.4.1

7.1.1 New SET LOGFILE Command

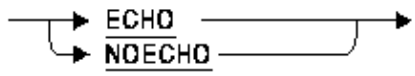
This release of Oracle Rdb adds a new SET LOGFILE statement to interactive SQL. This statement allows the executing SQL script to save output to an OpenVMS file.

Syntax

set-output=



logfile-options =



Arguments

- quoted−filespec
  A valid OpenVMS file specification. Output from interactive SQL will be written to this file.
- NOLOGFILE
  Closes the current output file specified by a prior SET LOGFILE (or SET OUTPUT command).
- NOOUTPUT
  Suspends writing to the output file.
- ECHO
  In addition to writing the output to the designated file, all commands and errors generated by interactive SQL are also written to SYS$OUTPUT.
- NOECHO
  Disable output to SYS$OUTPUT. All commands and errors generated by interactive SQL are only written to the output file.

Usage Notes

- SET LOGFILE is functionally equivalent to the SET OUTPUT statement. However, the SET OUTPUT statement is limited in functionality and is maintained for backward compatibility.

- Files opened with SET OUTPUT and SET LOGFILE can be subsequently processed by either a SET NOOUTPUT or a SET NOLOGFILE command.
- A SET LOGFILE command that does not specify a file is equivalent to SET NOLOGFILE.
- Output written by external functions, SQL TRACE statements, and other output enabled by the SET FLAGS command is never written to the SQL log file. Therefore, it cannot be captured using the statement.

Examples

Saving the output from a script

The following example shows the use of SET LOGFILE to save the output from a script without echoing the results.

1. The script being executed.

```
set verify;
start transaction read only;
set logfile (noecho) 'saved_date.log';
select rdb$flags from rdb$database;
set nologfile;
show alias;
rollback;
```

2. The output as seen during the Interactive SQL session.

```
SQL> start transaction read only;
SQL>
SQL> set logfile (noecho) 'saved_date.log';
SQL>
SQL> show alias;
Default alias:
    Oracle Rdb database in file SQL$DATABASE
SQL> rollback;
```

3. The output saved in the log file.

```
SQL>
SQL> select rdb$flags from rdb$database;
   RDB$FLAGS
           0
1 row selected
SQL>
SQL> set nologfile;
```

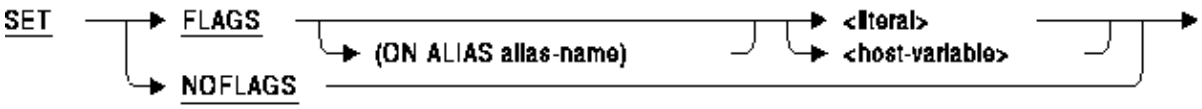
7.1.2 SET FLAGS Statement Now Allows ON ALIAS Clause

This release of Oracle Rdb extends the SET FLAGS statement to support an ON ALIAS clause. The default behavior for SET FLAGS is to establish the flag settings on all currently attached databases. This new clause will allow the database administrator to set flags on just one database alias.

The following example shows a case where enabling AUTO_OVERRIDE required DBADM privilege on the target database but not on the source database. It may be that the current user does not have (or really need) DBADM privilege on that database.

```
SQL> -- Now enable AUTO_OVERRIDE on only one database
SQL> set flags (on alias abc_a) 'auto_override';
SQL> set flags (on alias abc_b) 'none';
SQL> insert into abc_a.SAMPLE_TABLE select * from abc_b.SAMPLE_SOURCE;
SQL> commit;
```

Syntax



Arguments

- alias−name
  The name of an alias as declared by the ATTACH or CONNECT statement. If no ALIAS clause is used, then the alias name will default to RDB$DBHANDLE.

7.1.3 SQL Compiler−Generated Name Uniqueness Enhanced

Bug 4119771

In previous versions of Oracle Rdb, the SQL module language compiler (SQLMOD) and the SQL precompiler (SQLPRE) would generate object names based solely on the system time. This could, in some cases, result in duplicate names being generated for multiple different objects that were compiled at the same time by different processes.

This problem, for example, could cause linker warnings that show the symbol with a name similar in format to "SQL$PROC_1_A3DB62_3331BC" that had been created twice from within different object modules.

This problem has been corrected in Oracle Rdb Release 7.2.4.1. The SQL module language compiler and the SQL precompiler now create unique names within a system or a cluster. The names are comprised of a prefix, a request number and a string comprised of a cluster−wide unique value. The unique value includes components of the system time and the ID of the compiling process. The format of the new names is similar to "SQL$PRC9_DJHS2IHDBAA1G8BQ26A0".

7.1.4 Reduced CPU Usage and Improved Performance

Several performance enhancements have been implemented in this release of Oracle Rdb. Most of these changes are either specific to applications running on I64 systems or will have a greater effect on I64 systems. These enhancements include:

- Streamlined code sequences
- Reduced alignment faults

7.1.5 New RMU /SHOW STATISTICS /WRITE_REPORT_DELAY=n Feature

Bug 3199615

Previously, it was not possible to use the RMU /SHOW STATISTICS to write a report file in non−interactive mode.

This problem has been corrected in Oracle Rdb Release 7.2.4.1. The /WRITE_REPORT_DELAY=n qualifier specifies that statistics are to be collected for "n" seconds (default of 60 seconds) and then a report file written and then the RMU /SHOW STATISTICS utility will exit. /WRITE_REPORT_DELAY implies /NOINTERACTIVE.

Chapter 8
Enhancements And Changes Provided in Oracle Rdb Release 7.2.4

8.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.4

8.1.1 Date/Time Arithmetic Enhancements

Bug 6219485

This release of Oracle Rdb lifts many restrictions on the DATE VMS data type. SQL now treats DATE VMS type as a TIMESTAMP(2) for the purposes of the add and subtract with intervals or when generating intervals.

- A year−month interval can be added to or subtracted from a DATE VMS value and result in a DATE VMS value.
- A day−time interval can be added to or subtracted from a DATE VMS value and result in a DATE VMS value.
- A DATE VMS value can be subtracted from another DATE VMS value to produce either a year−month interval or a day−time interval.
- A DATE ANSI value can be subtracted from a DATE VMS value to produce a year−month interval.
- A DATE VMS value can be subtracted from a DATE ANSI value to produce either a year−month interval or a day−time interval.
- A DATE VMS value can be subtracted from a TIMESTAMP value to produce either a year−month interval or a day−time interval.
- A TIMESTAMP value can be subtracted from a DATE VMS value to produce either a year−month interval or a day−time interval.

Also in this release the following rules have been relaxed.

- Relax rule that TIME values could only be subtracted if they had the same fractional seconds precision.
- Relax rules that TIMESTAMP values could only be subtracted if they had the same fractional seconds precision.
- Relax assignment rules and let Rdb handle truncating time portion when TIMESTAMP is assigned to a DATE ANSI column, variable or parameter.
- Add rule that merge of DATE VMS and TIMESTAMP(n) will always be TIMESTAMP(n), where n is inherited from the TIEMSTAMP expression. Merge rules are used by UNION, INTERSECT, EXCEPT, MINUS operators and CASE expression processing.

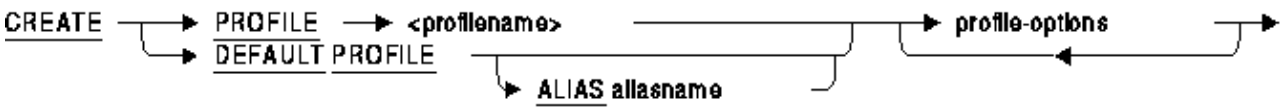
These enhancements have been made in Oracle Rdb Release 7.2.4.

8.1.2 New DEFAULT PROFILE Feature

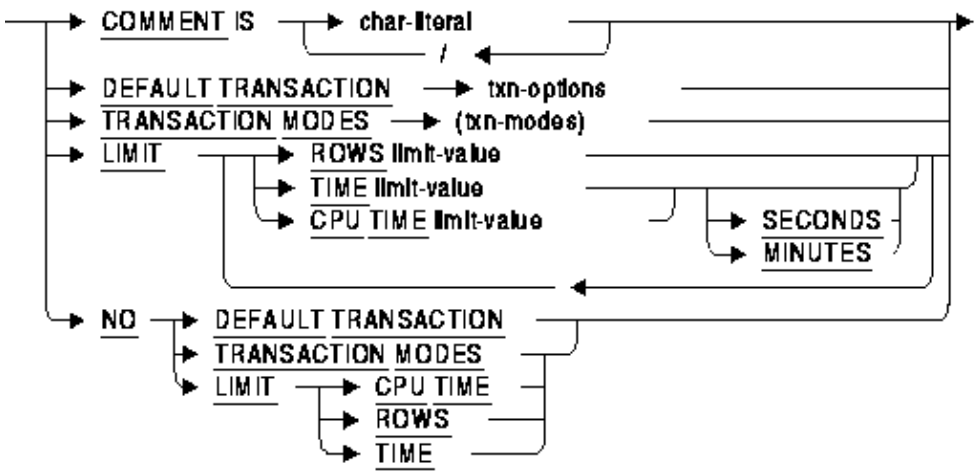
This release of Oracle Rdb enhances the PROFILE support with a new DEFAULT profile. When a user attaches to the database using ATTACH, CONNECT or SET SESSION AUTHORIZATION, they will either load their assigned profile definition or inherit the default profile (if defined).

Syntax

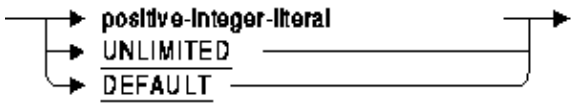
The CREATE, ALTER and DROP PROFILE syntax is changed as shown. The existing profile−options diagram remains the same.

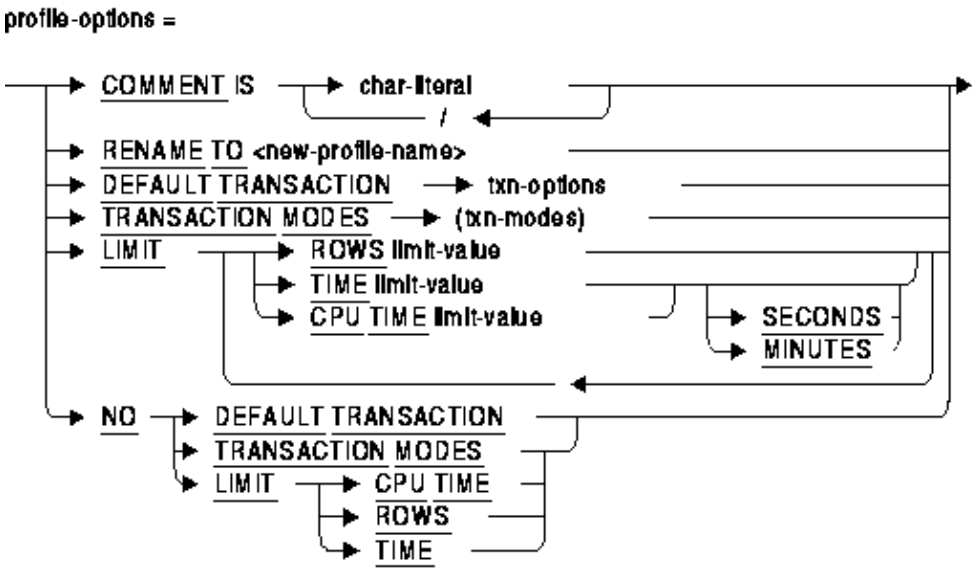


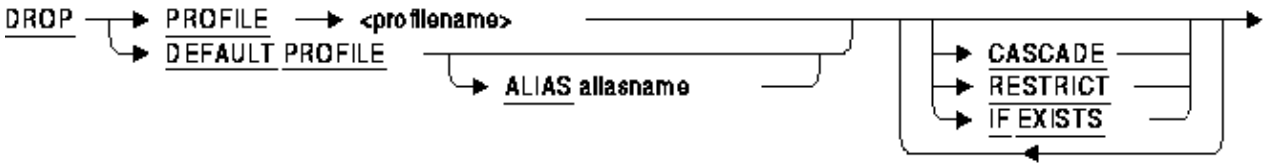
profile-options =



limit-value =







Arguments

- ALIAS aliasname
  When attached to multiple databases, the aliasname is required to direct the CREATE, ALTER or DROP command to the appropriate database.
- DEFAULT PROFILE
  Creates the special profile RDB$DEFAULT_PROFILE. This profile will be used by any user who is not assigned a profile using the PROFILE clause of CREATE or ALTER PROFILE.

Usage Notes

- It is possible to restrict the transaction modes to READ ONLY using the default profile. Use caution in this case because it is possible that no user will have READ WRITE access to undo such a definition. In this case, you can define the logical name RDMS$SET_FLAGS to the value PROFILE_OVERRIDE to allow a suitably privileged user to start a transaction without using the transaction mode restrictions in the default profile. Such a user must have database SECURITY privilege, possibly inherited from the OpenVMS SECURITY process privilege.

8.1.3 RMU /DUMP/BACKUP/OPTIONS=ROOT /HEADER_ONLY Displays the Header Information Only

Bug 8235615

A new feature has been added to RMU/DUMP/BACKUP/OPTIONS=ROOT command to process only the header information when the /HEADER_ONLY qualifier is used.

In prior releases of Oracle Rdb, the user had to wait until the entire backup file (.RBF) was processed. If the backup file was stored on tape and spanned multiple tapes then all the tapes had to be mounted and processed. When using /HEADER_ONLY, RMU now ceases processing of the backup file once the header has been displayed.

```
$ RMU/DUMP /BACKUP/OPTIONS=ROOT /HEADER_ONLY DBNODE$LMA200:GLORY.RBF
*-------------------------------------------------------------------------------
* Oracle Rdb V7.2-4                                        26-FEB-2009 07:21:58.69
*
* Dump of Database Backup Header
*     Backup filename: GLORY.RBF
*     Backup file database version: 7.2
*
*-------------------------------------------------------------------------------

Database Parameters:
    Root filename is "USER1:[BUG.8235615.FIX]GLORY.RDB;1"
    Created at 25-APR-2007 16:43:05.52
    Oracle Rdb structure level is 72.1
    Maximum user count is 23
    Maximum node count is 3
    Database open mode is AUTOMATIC
    Database close mode is AUTOMATIC
    Database will be mapped in process space
    All transaction modes are allowed
    Prestarted transactions are enabled
    Snapshot mode is NON-DEFERRED
    Statistics are enabled
    Operator notification is disabled
    Logical area count is 512
    Storage Areas...
      - Active storage area count is 4
      - Reserved storage area count is 8
    Row Caches...
      - Active row cache count is 0
      - Reserved row cache count is 1
      - Checkpoint information
          No time interval is specified
          Default source is updated rows
          Default target is backing file
          Default backing file directory is database directory
          RUJ Global Buffers are disabled
      - WARNING: Maximum node count is 3 instead of 1
      - WARNING: After-image journaling is disabled
      - WARNING: Fast commit is disabled
    Buffers...
      - Default user buffer count is 2000
      - Default recovery buffer count is 2000 (stored as 20)
      - Global buffers are disabled
          Global buffer count is 115
```

```
      Maximum global buffer count per user is 5
      Large memory is disabled
  - Buffer size is 12 blocks
      Maximum pages per buffer is 6
  - Asynchronous pre-fetch is enabled
      Maximum pre-fetch depth is 8 buffers
  - Detected asynchronous pre-fetch is enabled
      Maximum pre-fetch depth is 4 buffers
      Pre-fetch threshold is 4 buffers
  - Asynchronous batch-write is enabled
      Clean buffer count is 5
      Maximum batch size is 10 buffers
  - Optimized page transfer is disabled
Locking...
  - Adjustable record locking is enabled
      Fanout factor 1 is 10 (10 pages)
      Fanout factor 2 is 10 (100 pages)
      Fanout factor 3 is 10 (1000 pages)
  - Carry-over lock optimization is enabled
  - Lock tree partitioning is disabled
RUJ Journaling...
  - No default recovery-unit journal directory
AIJ Journaling...
  - After-image journaling is disabled
  - Database is configured for 7 journals
  - Reserved journal count is 7
  - Available journal count is 0
  - LogMiner is disabled
  - 7 journals can be created while database is active
  - Shutdown time is 60 minutes
  - Backup operation is manual
  - Default backup filename edits are not used
  - Log server startup is MANUAL
  - Journal overwrite is disabled
  - AIJ cache on "electronic disk" is disabled
  - Default journal allocation is 512 blocks
  - Default journal extension is 512 blocks
  - Default journal initialization is 512 blocks
  - Current roll-forward sequence number is 0
  - Current backup sequence number is 0
Fast Commit...
  - Fast commit is disabled
  - No checkpointing AIJ interval is specified
  - No checkpointing time interval is specified
  - No checkpointing transaction interval is specified
  - Commit to AIJ optimization is disabled
  - Transaction interval is 256
Hot Standby...
  - WARNING: After-image journaling is disabled
  - WARNING: Fast commit is disabled
  - WARNING: Log server startup is MANUAL
  - Informational: Operator notification is disabled
  - Database is not currently being replicated
Security Auditing...
  - Security auditing is disabled
  - Security alarm is disabled
  - No audit journal filename is specified
  - No alarm name is specified
  - Synchronous audit record flushing is disabled
  - Audit every access
Database Backup...
  - Fast incremental backup is enabled
```

```
      - Last full database backup was on 26-FEB-2009 07:16:56.09
      - Full database backup TSN is 0:128
      - Database was restored on 26-FEB-2009 06:52:11.82
    Derived Data...
      - Global section size
          With global buffers disabled is 277187 bytes (1MB)
          With global buffers enabled is 1036584 bytes (1MB)
            With Large memory global buffers enabled...
              Database TROOT section is 330024 bytes (1MB)
              Large memory global buffers section is 706560 bytes (1MB)
      - Row Cache RUJ buffers section size is 6041088 bytes (6MB)

Database root file ACL

(IDENTIFIER=[RDB,SFRANN],ACCESS=READ+WRITE+CONTROL+RMU$ALTER+RMU$ANALYZE+
     RMU$BACKUP+RMU$CONVERT+RMU$COPY+RMU$DUMP+RMU$LOAD+
     RMU$MOVE+RMU$OPEN+RMU$RESTORE+RMU$SECURITY+RMU$SHOW+RMU$UNLOAD+RMU$VERIFY)
```

8.1.4 GET ENVIRONMENT Now Supports SQLCODE and SQLSTATE Capture

This release of Oracle Rdb allows the GET ENVIRONMENT (SESSION) command to return the SQLCODE and SQLSTATE of the last executed statement. The keyword SQLCODE returns an INTEGER value and SQLSTATE returns a CHAR(5) value. The execution of the GET ENVIRONMENT statement will clear these values so both should be fetched in the same statement.

The following example shows a raised error and the use of GET ENVIRONMENT to capture the SQLCODE and SQLSTATE.

```
SQL> declare :st char(5);
SQL> declare :sc integer = -1;
SQL>
SQL> begin set :sc = :sc / 0; end;
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-COSI-F-ARITH, arithmetic exception
-COSI-F-FLTDIV, floating point division exception
SQL>
SQL> get environment (session) :st = SQLSTATE, :sc = SQLCODE;
SQL>
SQL> print :st, :sc;
 ST               SC
 22003          -304
SQL>
```

Once the SQLCODE or SQLSTATE value has been saved to a declared variable, it can be used to conditionally execute a COMMIT or a ROLLBACK.

```
begin
if :sc < 0 then
    rollback;
else
    commit;
end if;
end;
```

8.1.5 Timestamp Added to Messages For RMU LOAD and UNLOAD

In order to help judge progress of RMU LOAD and UNLOAD operations, a timestamp has been added to the RMU−I−DATRECSTO and RMU−I−DATRECUNL progress messages. The following example shows these timestamps.

```
$RMU /UNLOAD MFP C1 C1
%RMU-I-DATRECUNL,   200000 data records unloaded  5-JUN-2009 07:58:17.23.
$RMU /LOAD /COMMIT=75000 /LOG_COMMIT MFP C1 C1
%RMU-I-DATRECSTO,   75000 data records stored  5-JUN-2009 07:58:43.09.
%RMU-I-DATRECSTO,   150000 data records stored  5-JUN-2009 07:58:43.12.
%RMU-I-DATRECSTO,   200000 data records stored  5-JUN-2009 07:58:43.14.
```

8.1.6 New SET SQLDA Statement

Bugs 1088554, 4179408, 5414051, and 7022262

This release of Oracle Rdb introduces a new SET SQLDA statement.

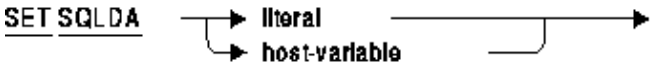
The SQLDA Statement allows a programmer using Dynamic SQL to alter the way the SQLDA (and SQLDA2) and Dynamic SQL statements are processed by Oracle Rdb.

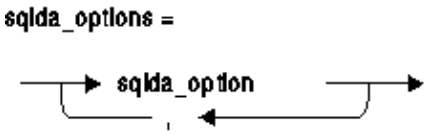
Environment

You can use the SET SQLDA statement:

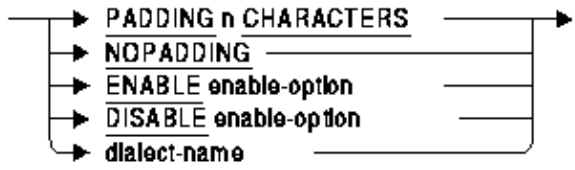
- In dynamic SQL as a statement to be dynamically executed

Syntax

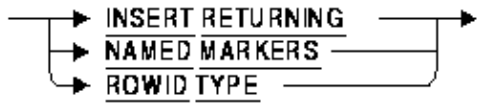




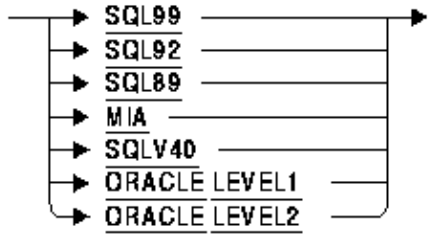
sqlda_option =



enable-option =



dialect-name =



Arguments

- literal
  host−variable
  The parameter passed to the statement must be a literal or a host variable containing one or more SQLDA options (see sqlda_options syntax diagram for details). If more than one option is specified, they must be separated by commas.
- sqlda_options
  One or more keyword clauses. If more than one clause is specified, they must be separated by commas.
- ENABLE
  The ENABLE clause activates one of the following behaviors for Dynamic SQL.
  - ♦ INSERT RETURNING − The default behavior of INSERT ... RETURNING when executed by dynamic SQL is to place parameters from the RETURNING INTO clause into the INPUT SQLDA. This behavior is maintained for backward compatibility. This option allows the programmer to force different (and correct) behavior for the non−compound use of this statement.

Note

If the INSERT RETURNING statement is included in a compound statement, the parameters are handled correctly.

- ♦ NAMED MARKERS − as well as traditional parameters markers (?). Dynamic SQL will now accept named, host variable style parameter markers. See the Usage Notes for further details and examples.
- ♦ ROWID TYPE − returns DBKEY values as a special type (SQLDA_ROWID, 455) to make processing of the DBKEY values easier. For instance, in prior releases, the SQLDA name field (SQLNAME) for DBKEY entries in the SQLDA was the only way to distinguish these values from other CHAR or VARCHAR columns − it would be either DBKEY or ROWID. If a query renamed the DBKEY column, then the application had no information in the SQLDA to indicate that the CHAR or VARCHAR value was binary data. In all respects, the SQLDA_ROWID type appears as a fixed length string of octets (possibly containing octets of zero which the C language would treat as a NULL terminator for a string).

- DISABLE
  The DISABLE clause deactivates one of the specified behaviors for Dynamic SQL. See ENABLE clause for a list of options.
- ORACLE LEVEL1
  ORACLE LEVEL2
  Either of these options will set the SQLDA to supply enhanced semantics. These options are currently reserved for use of the OCI Services for Rdb product that is part of the Oracle Rdb SQL/Services component. This setting also implicitly enables NAMED MARKERS.
- PADDING n CHARACTERS
  This option directs SQL to configure the SQLDA with larger CHARACTER VARYING strings than would normally be seen. The value of n is an unsigned numeric literal that specifies the number of characters that are added to the estimated length. Any CHARACTER (CHAR) types are converted to CHARACTER VARYING (VARCHAR). This rule is applied to comparison operators <, <=, >, >=, =, <>, and string functions (STARTING WITH, CONTAINING).
- NOPADDING
  This option sets the number of padding characters to 0. This also implies that derived CHARACTER (CHAR) types are not converted to CHARACTER VARYING (VARCHAR) when PADDING CHARACTERS is used.

Note

Oracle recommends that applications always check for SQLDA_CHAR and SQLDA_VARCHAR so that the correctly formatted data is made available to SQL.

  This is the default setting.
- SQL99
  SQL92
  MIA
  SQL89
  SQLV40
  Any of these options will revert to the default semantic for the SQLDA which includes disabling NAMED MARKERS.

Usage Notes

- The ORACLE LEVEL1 and ORACLE LEVEL2 settings are reserved for use by Oracle Corporation. Current behavior of this setting may change with any given release based on requirements of the OCI Services for Rdb component. This setting changes the usage of various SQLDA and SQLDA2 fields.
- Keywords may not be abbreviated and the clauses must be fully specified.
- The SET DIALECT command will implicitly enable NAMED MARKERS if the dialect is changed to either ORACLE LEVEL1 or ORACLE LEVEL2.
- The SET DIALECT command will implicitly disable NAMED MARKERS if the dialect is changed to any dialect other than ORACLE LEVEL1 or ORACLE LEVEL2.
- When NAMED MARKERS are enabled, the contents of the SQLDA and SQLDA2 will reflect one entry for each name. When traditional parameter markers are used, a SQLDA (or SQLDA2) entry will exist for each marker (?) encountered. This change in behavior can simplify the query encoding as well lead to more efficient strategy creation.

Examples

Using the NAMED MARKERS Feature

This example shows that enabling the NAMED MARKERS feature will allow SQL to prompt for one value and the displayed Rdb strategy shows that only one variable is used.

```
-> SET SQLDA 'ENABLE NAMED MARKERS';
-> SELECT LAST_NAME FROM EMPLOYEES WHERE FIRST_NAME = :F_NAME AND LAST_NAME <>
:F_NAME;
in:  [0] typ=449 len=46
out: [0] typ=453 len=14
[SQLDA - reading 1 fields]
-> Alvin
Tables:
  0 = EMPLOYEES
Conjunct: (0.FIRST_NAME = <var0> AND (0.LAST_NAME <> <var0>))
Get     Retrieval sequentially of relation 0:EMPLOYEES
 0/FIRST_NAME/Varchar(42/46): Alvin
[SQLDA - displaying 1 fields]
 0/LAST_NAME: Toliver
[SQLDA - displaying 1 fields]
 0/LAST_NAME: Dement
```

Using the PADDING Feature

The following example shows that the derived type for the named parameter MI is a SQLDA_CHAR (453) of length 1. The input data ('AA') is truncated on assignment and the incorrect results are returned. By adding a small padding, the type is changed to SQLDA_VARCHAR (449) of length 3 and a correct comparison is performed.

```
-> ATTACH 'filename sql$database';
-> SET SQLDA 'enable named markers, nopadding';
-> SELECT LAST_NAME FROM EMPLOYEES WHERE MIDDLE_INITIAL = :MI;
in:  [0] typ=453 len=1
out: [0] typ=449 len=18
[SQLDA - reading 1 fields]
-> AA
[SQLDA - displaying 1 fields]
 0/LAST_NAME: Toliver
[SQLDA - displaying 1 fields]
 0/LAST_NAME: Lengyel
[SQLDA - displaying 1 fields]
```

```
 0/LAST_NAME: Robinson
[SQLDA − displaying 1 fields]
 0/LAST_NAME: Ames
−> SET SQLDA 'padding 2 characters';
−> SELECT LAST_NAME FROM EMPLOYEES WHERE MIDDLE_INITIAL = :MI;
in:  [0] typ=449 len=7
out: [0] typ=449 len=18
[SQLDA − reading 1 fields]
−> AA
−> EXIT;
Enter statement:
```

Note that the VARCHAR requires an extra 4 bytes for the length information in the SQLDA2 used by the Dynamic SQL testing program.

8.1.7 RMU /SHOW VERSION Displays System Architecture and Version

The RMU /SHOW VERSION command has been enhanced to include information about the system architecture and OpenVMS version as shown in the following example:

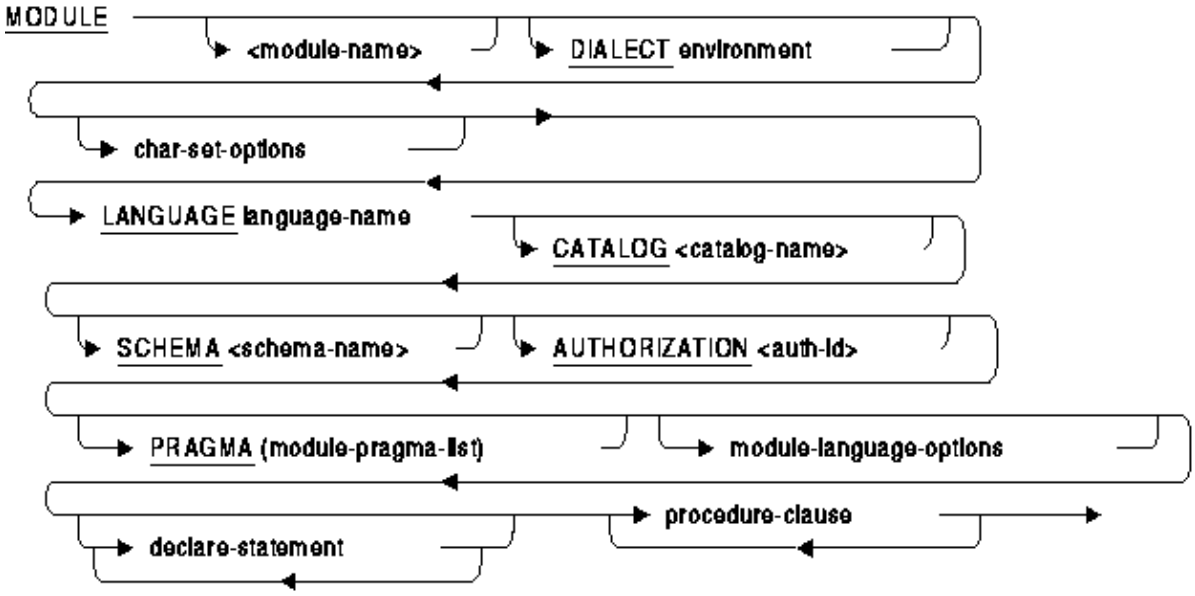
```
$ RMU /SHOW VERSION
Executing RMU for Oracle Rdb V7.2−400 on OpenVMS IA64 V8.3−1H1
$
```

8.1.8 New IDENT Option for SQL Module Language PRAGMA Clause

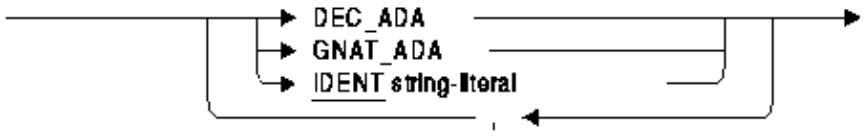
Many OpenVMS compilers allow the programmer to specify the object file IDENT string. This allows tracking of the correct version in linker map files (.map) and within object libraries using the LIBRARIAN command. This release of Oracle Rdb supports setting the IDENT string for the SQL module language.

The IDENT string is specified as part of the PRAGMA clause in the module header.

Syntax



module-pragma-list =



Usage Notes

- By using the PRAGMA clause with IDENT you can record an identification string in the object module generated by the SQL Module language. This IDENT string is recorded by the OpenVMS LINKER in the image itself and can be viewed in the generated MAP file, examined using ANALYZE/OBJECT, and by the LIBRARIAN command when the object module is stored in an object library.
- OpenVMS limits the IDENT string to a 15 octet string. If the string is longer than this (even with trailing spaces) then an error will be reported by the SQL Module Language compiler.
- If the IDENT clause is omitted, then the default version string will default to 'V1.0' as is the practice with many OpenVMS compilers. Prior versions of Oracle Rdb on Integrity systems would only provide the string '0'.

```
Module name:                            "MODSQL$TEST"
Module version:                         "0"
Creation date/time:                     "15-JUN-2009 20:13"
Language name:                          "Oracle Rdb SQL V7.2-351"
```

Examples

The following example shows the use of a PRAGMA clause in a module header to specify the module ident string.

Example 8−1 PRAGMA Clause in the Module Header

```
MODULE         MODSQL$TEST
DIALECT        SQL99
LANGUAGE       C
AUTHORIZATION  SAMPLE_USER
PRAGMA         (IDENT 'V1.2−300')
ALIAS          RDB$DBHANDLE
PARAMETER      COLONS
```

The DCL command ANALYZE/OBJECT can be used to examine the ident string in the object file.

Example 8−2 Examining the IDENT in the Object Module

```
$ sql$mod TEST
$ analyze/object TEST/interactive
This is an OpenVMS IA64 (Elf format) object file

Module Identification Information, in note section 2.

    Module name:                                "MODSQL$TEST"
    Module version:                             "V1.2−300"
    Creation date/time:                         "15−JUN−2009 20:04"
    Language name:                              "Oracle Rdb SQL V7.2−401"
Press RETURN to continue, or enter a period (.) for next file:
<Ctrl/Z>
$
```

Here is similar output from an OpenVMS Alpha system.

```
$ sql$mod TEST
$ analyze/object TEST
   .
   .
   .
This is an OpenVMS Alpha object file

1.  MODULE HEADER (EOBJ$C_EMH), 71 bytes

        structure level: 2
        maximum record size: 4088
        module name: "MODSQL$TEST"
        module version: "V1.0"
        creation   date/time: 16−JUN−2009 11:02
   .
   .
   .
```

This example shows the use of the LIBRARIAN to display the ident strings for object modules in a project object library.

```
$ librarian/list/full project.olb
Directory of ALPHA OBJECT library DISK1:[TESTER]PROJECT.OLB;1 on 16−JUN−2009
11:07:23
```

```
Creation date:  16-JUN-2009 11:07:11      Creator:  Librarian A09-30
Revision date:  16-JUN-2009 11:07:11      Library format:   3.0
Number of modules:      1                 Max. key length:  128
Other entries:          5                 Preallocated index blocks:    213
Recoverable deleted blocks:        0      Total index blocks used:        2
Max. Number history records:      20      Library history records:        0

MODSQL$TEST      Ident V1.2-300         Inserted 16-JUN-2009 11:07:11 5 symbols
```

8.1.9 New Keyword for SQL Module Language /PRAGMA Qualifier

This release of Oracle Rdb adds a new option to the /PRAGMA qualifier. The keyword IDENT can be used to pass a text string to the SQL Module Language compiler to be written to the Object Module Header.

The following example demonstrates the use of the qualifier to establish the generation of the compiler module.

```
$ SQL$MOD TEST/PRAGMA=IDENT="v1.2-32"
```

Usage Notes

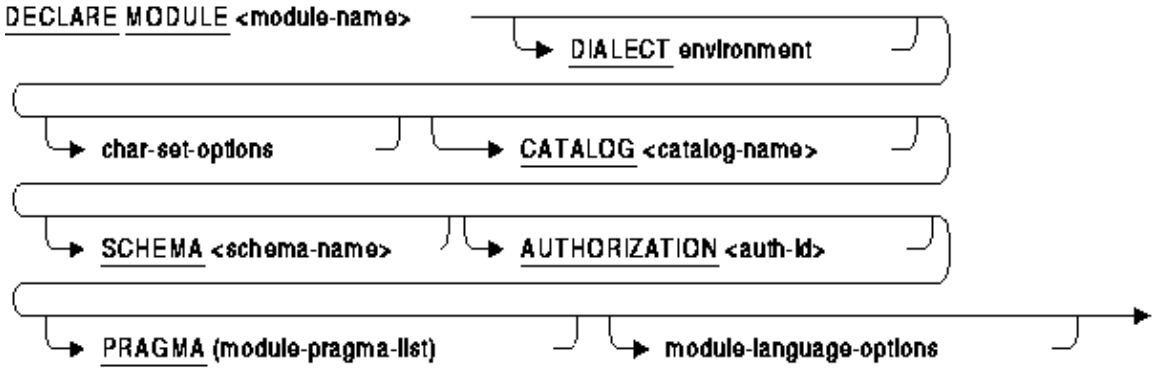
- If the PRAGMA (IDENT ...) clause is used as part of the MODULE header then that value will override any value used on the command line.
- The ANALYZE/OBJECT and LIBRARY commands can be used to display this IDENT string and the value will be displayed in LINKER map files.
- OpenVMS limits the IDENT string to a 15 octet string. If the string is longer than this (even with trailing spaces) then an error will be reported by the SQL Module Language compiler.

8.1.10 New IDENT Option for SQL Precompiler DECLARE MODULE Statement

Many OpenVMS compilers allow the programmer to specify the object file IDENT string. This allows tracking of the correct version in linker map files (.map) and within object libraries using the LIBRARIAN command. This release of Oracle Rdb supports setting the IDENT string for the SQL precompiler module header.

The IDENT string is specified as part of the PRAGMA clause in the module header.

Syntax



module-pragma-list =

Usage Notes

- By using the PRAGMA clause with IDENT, you can record an identification string in the object module generated by the SQL Precompiler. This IDENT string is recorded by the OpenVMS LINKER in the image itself and can be viewed in the generated MAP file, examined using ANALYZE/OBJECT, and by the LIBRARIAN command when the object module is stored in an object library.
- OpenVMS limits the IDENT string to a 15 octet string. If the string is longer than this (even with trailing spaces), then an error will be reported by the SQL Precompiler.
- If the IDENT clause is omitted, then the default version string will default to 'V1.0' as is the practice with many OpenVMS compilers. Prior versions of Oracle Rdb on Integrity systems would only provide the string '0'.

```
Module name:                            "MODSQL$TEST"
Module version:                         "0"
Creation date/time:                     "15-JUN-2009 20:13"
Language name:                          "Oracle Rdb SQL V7.2-351"
```

Examples

The following example shows the use of a PRAGMA clause in a module header to specify the module ident string.

Example 8-3 PRAGMA clause in the DECLARE MODULE statement

```
$ CREATE TEST_HDR.SQL
DECLARE
    MODULE          MODSQL$TEST
    DIALECT         SQL99
    AUTHORIZATION   SAMPLE_USER
    PRAGMA          (IDENT 'V1.2-300')
```

```
;
```

The DCL command ANALYZE/OBJECT can be used to examine the ident string in the object file. Note that the SQL Precompiler generates two object files which are concatenated. Therefore, the ANALYZE will show two MODULE HEADER records, one for the host language (C for example) and one from the SQL Precompiler.

Example 8−4 Examining the IDENT in the object module

```
$ sql$pre/CC TEST TEST_HDR
$ analyze/object TEST/output=SYS$OUTPUT
   .
   .
   .
This is an OpenVMS Alpha object file

175.  MODULE HEADER (EOBJ$C_EMH), 75 bytes
        structure level: 2
        maximum record size: 4088
        module name: "MODSQL$TEST"
        module version: "V1.2−300"
        creation   date/time: 16−JUL−2009 16:50
   .
   .
   .
```

8.1.11 New Keyword for SQL Precompiler PRAGMA Option

This release of Oracle Rdb adds a new keyword to the SQLOPTIONS qualifiers PRAGMA option. The keyword IDENT can be used to pass a text string to the SQL Precompiler to be written to the Object Module Header.

The following example demonstrates the use of the qualifier to establish the generation of the compiler module.

```
$ SQL$PRE/CC TEST/SQLOPTION=(PRAGMA=IDENT="v1.2−32")
```

Usage Notes

- If the PRAGMA (IDENT ...) clause is used as part of the DECLARE MODULE statement, then that value will override any value used on the command line.
- The ANALYZE/OBJECT and LIBRARY commands can be used to display this IDENT string and the value will be displayed in LINKER map files.
- OpenVMS limits the IDENT string to a 15 octet string. If the string is longer than this (even with trailing spaces) then an error will be reported by the SQL precompiler.

8.1.12 RDB_STATS_DATABASE Example Program

Accessing performance information in a tabular fashion for Oracle Rdb databases can often be beneficial. In

particular, stored RMU /SHOW STATISTICS rate information in a database can be utilized to do trend anaysis and historical review of performance indicators.

RDB_STATS_DATABASE is a sample program that reads an RMU /SHOW STATISTICS binary file and converts all statistic values for each sample into a current rate per second. The statistics values are written to a database table named RMU$STATISTICS. If the RMU$STATISTICS table does not exist in the database, it will be created.

To use the RDB_STATS_DATABASE program, create a foreign command symbol with a value of "SQLSAMPLE:RDB_STATS_DATABASExx.EXE" (where xx is the version of Rdb) and pass an output database and an input binary statistics file name. The following example command sequence demonstrates one possible way that statistics can be gathered for one hour and then formatted.

```
$ RDB_STATS_DATABASE := $SQL$SAMPLE:RDB_STATS_DATABASE72
$ RMU /SHOW STATISTICS MFP -
        /NOINTERACTIVE -
        /OUTPUT = 2008-11-16-00-56.STATS -
        /UNTIL = "16-NOV-2008 11:00:00" -
        /TIME = 15
$ RDB_STATS_DATABASE MYDB.RDB 2008-11-16-00-56.STATS
```

This program can be used to capture either "static" data (from a perviously collected binary file) or "real time" data where records are written to the database as they are produced from RMU /SHOW STATISTICS, as in the following example (note that the RDB_STATS_DATABASE example program should be modified when used in this fashion to commit after every record):

```
$ RDB_STATS_DATABASE := $SQL$SAMPLE:RDB_STATS_DATABASE72
$ CREATE /MAILBOX RDB_STATS_DATABASE$MAILBOX -
    /PERMANENT -
    /LOG -
    /BUFFER_SIZE = 65535 -
    /MESSAGE_SIZE = 10000
$ SPAWN RMU /SHOW STATISTICS MFP -
        /NOINTERACTIVE -
        /OUTPUT = RDB_STATS_DATABASE$MAILBOX:
        /UNTIL = "16-NOV-2009 11:00:00" -
        /TIME = 60
$ RDB_STATS_DATABASE MYDB.RDB RDB_STATS_DATABASE$MAILBOX:
```

This example is intended solely to be used as a template for writing your own program. No support for this example template program is expressed or implied.

Oracle Corporation assumes no responsibility for the functionality, correctness or use of this example program. Oracle Corporation reserves the right to change the format and contents of the Oracle Rdb RMU SHOW STATISTICS binary output file at any time without prior notice.

The RDB_STATS_DATABASE example program is comprised of the following source modules found in SQL$SAMPLE:

- RDB_STATS_DATABASExx.C
- RDB_STATS_DATABASE_SQL1_xx.SQLMOD
- RDB_STATS_DATABASE_SQL2_xx.SQLMOD

Compile and link the RDB_STATS_DATABASE example program as follows:

```
$ CC /FLOAT=IEEE RDB_STATS_DATABASExx.C
$ SQL$MOD /FLOAT=IEEE RDB_STATS_DATABASE_SQL1_xx.SQLMOD
$ SQL$MOD /FLOAT=IEEE RDB_STATS_DATABASE_SQL2_xx.SQLMOD
$ LINK RDB_STATS_DATABASExx+−
    RDB_STATS_DATABASE_SQL1_xx+−
    RDB_STATS_DATABASE_SQL2_xx+−
    SQL$USER /LIBRARY
```

8.1.13 RCS Time−Based Cache Sweeping

Previously, the Record Cache Server (RCS) process would perform modified row cache "sweep" operations only when a cache was full (also known as "clogged") with modified rows. Now a database may be configured to perform timed cache sweeps. This feature is intended to help perform "lazy" updates of modified rows to the database from caches without performing a full cache checkpoint operation.

The timer for the periodic cache sweeps is specified with the "SWEEP INTERVAL is numeric−literal seconds" clause of the ALTER DATABASE ... ROW CACHE IS ENABLED statement, as in the following example:

```
ALTER DATABASE FILENAME MF_PERSONNEL
  ROW CACHE IS ENABLED (SWEEP INTERVAL IS 300 SECONDS);
```

The number of slots per cache to sweep is specified with the ALTER CACHE statement. Legal values for "SWEEP INTERVAL" are from 0 seconds (to disable periodic timed sweeps) to 3600 seconds (1 hour).

The RMU /SET ROW_CACHE command accepts a /[NO]SWEEP_INTERVAL=n qualifier as an alternate method to specify the periodic cache sweep timer. /NOSWEEP_INTERVAL disables periodic timed sweeps and /SWEEP_INTERVAL=n can be used to set the timer for the periodic cache sweeps. Legal values for /SWEEP_INTERVAL=n are from 1 second to 3600 seconds (1 hour).

The Record Cache Server (RCS) process log file contains information about periodic row cache "sweep" operations and can be a useful analysis tool.

Default Value

The intended default value for the SWEEP INTERVAL in a database is zero seconds (meaning disabled). It is, however, possible for a database that had originally been created with Oracle Rdb Release 7.0 to have a non−zero value. Customers using the row cache feature are advised to explicitly set the SWEEP INTERVAL parameter to either zero (to disable periodic timed sweeps) or the desired sweep interval period on all databases after upgrading to Release 7.2.4.

Use RMU /SET ROW_CACHE /NOSWEEP_INTERVAL

In Oracle Rdb Release 7.2.4, the "SWEEP INTERVAL is 0 seconds" clause of the ALTER DATABASE ... ROW CACHE IS ENABLED statement may not disable the periodic row cache "sweep" operations. Oracle recommends using the RMU /SET ROW_CACHE /NOSWEEP_INTERVAL command as an alternative. This problem will be corrected in a future release.

Incomplete Display Support

Oracle Rdb Release 7.2.4 does not include complete support for showing a database's periodic row cache "sweep" operation timer value. As a workaround prior to the next Oracle Rdb release, the RMU/DUMP/HEADER/OPTIONS=DEBUG command can be used to display the database parameter RCS_SWEEP_INTERVAL as in the following example:

```
$ RMU/DUMP/HEADER/OPTIONS=DEBUG/OUTPUT=X.X MF_PERSONNEL
$ SEARCH X.X RCS_SWEEP_INTERVAL
RCACHE_CNT = 11.          RCACHE_VBN = 153.         RCS_SWEEP_INTERVAL = 123.
```

8.1.14 RMU Command TSN Keyword and Qualifier Value

RMU commands that accept a TSN keyword or qualifier value now accept input formats as follows:

- A decimal string representing a quadword TSN value
- A hexadecimal string starting with "%X" representing a quadword TSN value
- A two−part decimal string separated by a colon representing a quadword TSN value as high and low longwords

Following are some example uses of input TSN values:

```
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=54321
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=123456234253245
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=%X7655
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=%X000000715F856AB
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=0:871251
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=3:53487
$ RMU /DUMP /AFTER_JOURNAL J1.AIJ /FIRST=TSN=21:653156
```

8.1.15 New Support for RENAME and CREATE SYNONYM Commands

With this release of Oracle Rdb, the RENAME and CREATE SYNONYM commands support INDEX and STORAGE MAP database objects.

- RENAME INDEX changes the name of the index in all system tables.
  A synonym is created using the old index name to reference the new name of the index. This synonym will be used by any query outline that previously referenced the index using the old name. Note that only a single synonym name may exist. Therefore, if you have indices with the same name as another object, then the RENAME INDEX command may fail if creating the synonym detects a duplicate name,
  The command ALTER INDEX ... RENAME TO ... is synonymous with the RENAME INDEX command.
- RENAME STORAGE MAP changes the name of the storage map in all system tables.

If the storage map has a companion function in the RDB$STORAGE_MAPS system module, then that function will also be renamed. A synonym is created using the old function name to reference the new name of the function. This synonym will be used by any other routine, computed by column, automatic column, and so on that referenced the old storage mapping function.
The command ALTER STORAGE MAP ... RENAME TO ... is synonymous with the RENAME STORAGE MAP command.

- CREATE SYNONYM ... FOR INDEX ... is now supported. Synonyms for indices can be created, altered and dropped.
- CREATE SYNONYM ... FOR STORAGE MAP ... is now supported. Synonyms for storage maps can be created, altered and dropped.

The following example shows the result of the RENAME INDEX and RENAME STORAGE MAP commands.

```
SQL> show table (storage maps,index) employees
Information for table EMPLOYEES

Indexes on table EMPLOYEES:
EMPLOYEES_HASH                  with column EMPLOYEE_ID
  No Duplicates allowed
  Type is Hashed Scattered
  Key suffix compression is DISABLED

EMP_EMPLOYEE_ID                 with column EMPLOYEE_ID
  No Duplicates allowed
  Type is Sorted
  Key suffix compression is DISABLED
  Node size  430

EMP_LAST_NAME                   with column LAST_NAME
  Duplicates are allowed
  Type is Sorted
  Key suffix compression is DISABLED

Storage Map for table EMPLOYEES:
     EMPLOYEES_MAP

SQL> rename storage map EMPLOYEES_MAP to EMP_STORAGE_MAP;
SQL> rename index EMPLOYEES_HASH to EMP_ID_HASH;
SQL> show table (storage maps,index) employees
Information for table EMPLOYEES


Indexes on table EMPLOYEES:
EMP_EMPLOYEE_ID                 with column EMPLOYEE_ID
  No Duplicates allowed
  Type is Sorted
  Key suffix compression is DISABLED
  Node size  430

EMP_ID_HASH                     with column EMPLOYEE_ID
  No Duplicates allowed
  Type is Hashed Scattered
  Key suffix compression is DISABLED

EMP_LAST_NAME                   with column LAST_NAME
  Duplicates are allowed
  Type is Sorted
  Key suffix compression is DISABLED
```

```
Storage Map for table EMPLOYEES:
     EMP_STORAGE_MAP

SQL> show storage map
User Storage Maps in database with filename mf_personnel_sql
     CANDIDATES_MAP
     COLLEGES_MAP
     DEGREES_MAP
     DEPARTMENTS_MAP
     EMP_STORAGE_MAP
     JOBS_MAP
     JOB_HISTORY_MAP
     SALARY_HISTORY_MAP
     WORK_STATUS_MAP
SQL> show index
User indexes in database with filename mf_personnel_sql
     COLL_COLLEGE_CODE
     DEG_COLLEGE_CODE
     DEG_EMP_ID
     DEPARTMENTS_INDEX
     EMP_EMPLOYEE_ID
     EMP_ID_HASH
     EMP_LAST_NAME
     JH_EMPLOYEE_ID
     JOB_HISTORY_HASH
     SH_EMPLOYEE_ID
     EMPLOYEES_HASH                  A synonym for index EMP_ID_HASH
SQL> show system function
Functions in database with filename mf_personnel_sql
     CANDIDATES_MAP
     COLLEGES_MAP
     DEGREES_MAP
     DEPARTMENTS_MAP
     EMP_STORAGE_MAP
     JOBS_MAP
     JOB_HISTORY_MAP
     SALARY_HISTORY_MAP
     WORK_STATUS_MAP
     EMPLOYEES_MAP                   A synonym for function EMP_STORAGE_MAP
SQL>
```

Chapter 9
Documentation Corrections, Additions and Changes

This chapter provides corrections for documentation errors and omissions.

9.1 Documentation Corrections

9.1.1 Oracle Rdb Release 7.2.x.x New Features Document Added

A new document has been created which contains all of the New Features Chapters from all previous Rdb 7.2 Release Notes. This document will be included in saveset A of the Rdb kit. It is called RDB_NEWFEATURES_72xx 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 7.2 releases.

9.1.2 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 their 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.

9.1.3 ROUND and TRUNC Are Built In Functions for SQL

The functions ROUND and TRUNC for numeric values are now supported as native functions in Oracle Rdb.

Fixed point values are now truncated and rounded correctly. Floating values, while supported by ROUND and TRUNC, may not always return the expected results. Please review usage of ROUND in such contexts.

The result type for ROUND and TRUNC will match the data type of the input source parameter.

Usage Notes

- The implementation of ROUND and TRUNC for DATE values requires the use of the OCI Services for Rdb library (also know as SQL*net for Rdb). These functions will now accept DATE ANSI, TIMESTAMP and DATE VMS values.
  Attempts to use ROUND or TRUNC on a database that is not setup for OCI Services will receive

errors similar to these:

```
SQL> select TRUNC (current_date) from rdb$database;
%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-BAD_SYM, unknown routine symbol - TRUN2
SQL> select ROUND (current_date) from rdb$database;
%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-BAD_SYM, unknown routine symbol - ROUN2
```

Note

The special functions ROUN2 and TRUN2 are internal routines to deal with DATE types.

- Both ROUND and TRUNC support the data types REAL, FLOAT and DOUBLE PRECISION for both parameter and results. However, due to the imprecise nature of floating point arithmetic, this may cause unexpected results. A value such as 4.185 will not round to 4.19 as expected because the internal (and approximate) representation of the number is something like 4.184999942780E+000 and therefore does not appear to require rounding to the second decimal place according to the rounding rules.
  The following example shows this problem.

```
SQL> select cast(round (4.185,2) as integer(2)) from rdb$database;

          4.18
1 row selected
SQL> select cast(round (4.175,2) as integer(2)) from rdb$database;

          4.18
1 row selected
SQL>
```

Note

The result of a divide operation (/) or the AVG, STDDEV, VARIANCE statistical functions are floating point values so applying TRUNC or ROUND to those results, even if performed on integer sources, will also be affected by the intermediate floating point type.

- If you use SQL to access older versions of Rdb (such as via remote access) then SQL will revert to the previous behavior and use the SQL functions provided by the SQL_FUNCTIONS library.

9.1.4 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>
```

9.1.5 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
```

9.1.6 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 − Specify the per−database default backing store location.
- NoBacking_Store_Location − Remove the per−database default backing store location and revert 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 processes working set as needed.
  - ♦ TYPE=RESIDENT to specify that the database global section is memory resident in process (P0) address space using OpenVMS Alpha 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])
```

9.1.7 Documentation for the DEBUG_OPTIONS Qualifier of RMU Unload

Bug 8447357

The RMU Help file and RMU Reference Manual is 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 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: 2009-04-23 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.

9.1.8 SQL$MSGxx.DOC Is Not Alphabetical

Bug 4387383

The last paragraph of page A−3 of volume 5 of the SQL Reference Manual says the message codes in files such as SQL$MSG71.DOC are alphabetized. However, it was found that the message codes were not alphabetized in SQL$MSG71.DOC or SQL$MSG72.DOC.

The cause of this problem was that the COSI message codes were appended to the end of the SQL message codes in this file.

This has been corrected in Oracle Rdb Release 7.2.4. We no longer append the COSI message codes to the SQL$MSGnn.DOC file since the COSI message codes are available separately.

9.1.9 LOCK_TIMEOUT Documentation Error in RMU Reference Manual Release 7.2

The "Oracle Rdb for OpenVMS: Oracle RMU Reference Manual Release 7.2" incorrectly implies that there is a default value for the lock timeout in seconds specified by the /LOCK_TIMEOUT qualifier in the following sections:

- 1.10 RMU/BACKUP COMMAND
- 1.11 RMU/BACKUP/AFTER_JOURNAL COMMAND
- 1.17 RMU COPY_DATABASE COMMAND

In all these sections, in the description of the "/Lock_Timeout=n" qualifier, any reference to a default value such as "The default value for the /Lock_Timeout=n qualifier is ..." needs to be removed since there is no default value allowed for this qualifier. If you specifiy the /LOCK_TIMEOUT qualifier, you have to specify the lock timeout value in seconds. If you do not specifiy the /LOCK_TIMEOUT qualifier, the default is to wait indefinitely to acquire the QUIET POINT lock and any other locks needed for ONLINE execution of the command. It should also be mentioned that the LOCK_TIMEOUT value does not only affect the QUIET POINT lock but can affect other locks RMU may need to acquire for ONLINE execution.

9.1.10 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>
```

9.1.11 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.

9.1.12 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.

9.1.13 Missing Example for CREATE STORAGE MAP

Bug 5655348

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

Example

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

```
SQL> create database
cont>     filename 'DB$:MULTIMEDIA'
cont>
cont>     create storage area PHOTO_AREA1
cont>         filename 'DB$:PHOTO_AREA1'
cont>         page format UNIFORM
cont>
cont>     create storage area PHOTO_AREA2
cont>         filename 'DB$:PHOTO_AREA2'
cont>         page format UNIFORM
cont>
cont>     create storage area TEXT_AREA
cont>         filename 'DB$:TEXT_AREA'
cont>         page format UNIFORM
cont>
cont>     create storage area AUDIO_AREA
cont>         filename 'DB$:AUDIO_AREA'
cont>         page format UNIFORM
cont>
cont>     create storage area DATA_AREA
cont>         filename 'DB$:DATA_AREA'
cont>         page format UNIFORM
cont> ;
SQL>
SQL> create table EMPLOYEES
cont>     (name       char(30),
cont>      dob        date,
cont>      ident      integer,
cont>      photograph list of byte varying (4096) as binary,
cont>      resume     list of byte varying (132) as text,
cont>      review     list of byte varying (80) as text,
cont>      voiceprint list of byte varying (4096) as binary
cont>     );
SQL>
SQL> create storage map EMPLOYEES_MAP
cont>     for EMPLOYEES
cont>     enable compression
cont>     store in DATA_AREA;f
SQL>
SQL> create storage map LISTS_MAP
cont>     store lists
cont>         in AUDIO_AREA
cont>                 (thresholds are (89, 99, 100)
cont>                 ,comment is 'The voice clips'
cont>                 ,partition AUDIO_STUFF)
cont>             for (employees.voiceprint)
cont>         in TEXT_AREA
cont>                 (thresholds is (99)
cont>                 ,partition TEXT_DOCUMENTS)
cont>             for (employees.resume, employees.review)
cont>         in (PHOTO_AREA1
```

```
cont>                (comment is 'Happy Smiling Faces?'
cont>                ,threshold is (99)
cont>                ,partition PHOTOGRAPHIC_IMAGES_1)
cont>             ,PHOTO_AREA2
cont>                (comment is 'Happy Smiling Faces?'
cont>                ,threshold is (99)
cont>                ,partition PHOTOGRAPHIC_IMAGES_2)
cont>             )
cont>             for (employees.photograph)
cont>             fill randomly
cont>         in RDB$SYSTEM
cont>                (partition SYSTEM_LARGE_OBJECTS);
SQL>
SQL> show storage map LISTS_MAP;
     LISTS_MAP
 For Lists
 Store clause:          STORE lists
        in AUDIO_AREA
                (thresholds are (89, 99, 100)
                ,comment is 'The voice clips'
                ,partition AUDIO_STUFF)
            for (employees.voiceprint)
        in TEXT_AREA
                (thresholds is (99)
                ,partition TEXT_DOCUMENTS)
            for (employees.resume, employees.review)
        in (PHOTO_AREA1
                (comment is 'Happy Smiling Faces?'
                ,threshold is (99)
                ,partition PHOTOGRAPHIC_IMAGES_1)
            ,PHOTO_AREA2
                (comment is 'Happy Smiling Faces?'
                ,threshold is (99)
                ,partition PHOTOGRAPHIC_IMAGES_2)
            )
            for (employees.photograph)
            fill randomly
        in RDB$SYSTEM
                (partition SYSTEM_LARGE_OBJECTS)

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

```
   Storage Area: RDB$SYSTEM
SQL>
SQL> commit;
```

9.1.14 RDM$BIND_MAX_DBR_COUNT Documentation Clarification

Bugs 1495227 and 3916606

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

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

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

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

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

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

Per−Database Value

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

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

9.1.15 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 9−1.

Table 9−1 Server Process Priority Logical Names

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

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

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

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

Bug 2500594

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

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

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

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

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

For example:

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

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

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

Now run SQL and verify that the version is correct:

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

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

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

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

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

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

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

9.1.17 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.

9.1.18 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.

9.1.19 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 9−2 shows the TRANS_TPB table.

Table 9−2 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 9−3 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 9−3 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)	

9.1.20 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 9−4 shows the DATABASE table.

Table 9−4 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 9−5 shows the REQUEST_ACTUAL table.

Table 9−5 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 9−6 shows the TRANSACTION table.

Table 9−6 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 9−7 shows the REQUEST_BLR table.

Table 9−7 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

9.2 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
```

9.3 Online Document Format and Ordering Information

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

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

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

Customers should contact their Oracle representative to purchase printed documentation.

Chapter 10
Known Problems and Restrictions

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

10.1 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.

10.1.1 Session Crash if Run Time Routine Native Compiler Enabled

Bug 13495444

A session may crash when using the Interpreter Compiler on Itanium (SET FLAGS 'CODE_OPTIMIZATION(2)'), which is the default, in some rare instances where a large single code routine is generated because of a long complicated query segment (for example, a long arithmetic expression). When executing the generated code, a stack corruption can occur resulting in the current process being deleted.

Turning off the Interpreter Compiler by using SET FLAGS 'CODE_OPTIMIZATION(0)' is a viable workaround.

10.1.2 Possible Incorrect Results When Using Partitioned Descending Indexes

Bug 6129797

In the current release of Oracle Rdb, 7.2.4, 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 during the index partitions.

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

10.1.3 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.

10.1.4 Case Sensitive Values in RDB$CLIENT_DEFAULTS.DAT

Bug 7681548

Various characteristics for network access to remote databases can be specified by entering parameters and values in a file named RDB$CLIENT_DEFAULTS.DAT. The following keywords that have character strings as their values only take effect if the values are specified in UPPER CASE: SQL_NETWORK_TRANSPORT_TYPE, SQL_MESSAGE_VECTOR_RETURN_TYPE, and SQL_DEFAULTS_RESTRICTION. The result of including one or more lower case characters in the value of one of these parameters is the same as if the parameter was not specified at all (for example, the default behavior would be applied and no error message would be issued).

In the following example, DECnet is specified with the last three characters in lower case. The result will be that the value of the parameter SQL_NETWORK_TRANSPORT_TYPE will be DEFAULT and not the intended value DECNET.

```
SQL_NETWORK_TRANSPORT_TYPE DECnet
```

This problem can be avoided by specifying the values for SQL_NETWORK_TRANSPORT_TYPE, SQL_MESSAGE_VECTOR_RETURN_TYPE, and SQL_DEFAULTS_RESTRICTION in RDB$CLIENT_DEFAULTS.DAT using UPPER CASE.

In the next major release of Oracle Rdb, the values in RDB$CLIENT_DEFAULTS.DAT will be case insensitive.

10.1.5 Standalone WITH Clause in Compound Statements Now Deprecated

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

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

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

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

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

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

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

10.1.6 Calling DECC$CRTL_INIT

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

DECC$CRTL_INIT is a C run time library routine that allows developers to call the HP C RTL from other languages or to use the HP C RTL when the main function is not in C. It initializes the run−time environment and establishes both an exit and condition handler. The Oracle Rdb main images are not written in C and should not be expected to have called DECC$CRTL_INIT prior to the user's code being invoked. The

requirement for DECC$CRTL_INIT in certain cases exists in all versions of Oracle Rdb.

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

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

10.1.7 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.

10.1.8 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.

10.1.9 Possible Incorrect Results When Using Partitioned Descending Indexes on I64

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

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

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

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

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

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

10.1.10 Changes for Processing Existence Logical Names

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

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

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

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

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

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

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

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

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

Bug 2351258

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

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

```
DECLARE MANL_NAME_LIST CURSOR FOR
```

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

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

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

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

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

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

- TO_CHAR
- TO_NUMBER
- TO_DATE

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

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

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

10.1.15 Partitioned Index with Descending Column and Collating Sequence

Bug 2797443

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

The following example can be used to demonstrate the problem.

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

set flags 'strategy';

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

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

! Here is the same query without the sequential access restriction.
! Note in the query strategy that index Y3 is used for data retrieval.
```

```
! This query ought to (but does not) return the same set of rows as
! for the sequential access query.

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

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

Bug 3735144

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

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

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

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

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

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

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

There are three potential workarounds:

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

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

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

10.1.17 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.

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

Bug 2676851

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

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

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

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

10.1.19 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.

10.1.20 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.

10.1.21 Optimization of Check Constraints

Bug 1448422

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

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

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

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

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

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

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

```
SQL> alter table t1 alter column f1
```

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

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

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

or:

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

In both cases the retrieval strategy will look like this:

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

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

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

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

```
SQL> create trigger t1_insert after insert on t1
cont>  when (not exists (select * from t2 where f2=f1))
cont>    (error) for each row;
SQL> create trigger t1_update after update on t1
cont>  when (not exists (select * from t2 where f2=f1))
cont>    (error) for each row;
SQL> ! A delete trigger is not needed on T1.
```

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

The strategy for a delete on T2 is now:

```
SQL> delete from t2 where f2=1;
Aggregate-F1    Index only retrieval of relation T1
  Index name  I1 [1:1]
Temporary relation      Get     Retrieval by index of relation T2
  Index name  I2 [1:1]
%RDB-E-TRIG_INV_UPD, invalid update; encountered error condition defined for
trigger
-RDMS-E-TRIG_ERROR, trigger T2_DELETE forced an error
```

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

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

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

10.1.22 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.

10.1.23 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.

10.1.24 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.

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

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

Oracle Rdb includes its own copy of the OpenVMS SORT32 code within the Oracle Rdb images and does not generally call the routines in the OpenVMS run−time library. A copy of the SORT32 code is used to provide stability between versions of Oracle Rdb and OpenVMS and because Oracle Rdb calls the sort routines from

executive processor mode which is difficult to do using the SORT32 shareable image. SQL IMPORT and RMU Load operations do, however, call the OpenVMS SORT run−time library.

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

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

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

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

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

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

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

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

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

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

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

10.1.26 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 10−1 Sort Memory Logicals

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

10.1.27 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.

10.2 SQL Known Problems and Restrictions

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

10.2.1 SET FLAGS CRONO_FLAG Removed

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

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

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

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

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

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

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

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

This restriction does not apply to interactive or dynamic SQL.

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

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

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

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

10.2.4 Multistatement or Stored Procedures May Cause Hangs

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

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

```
Session 1:

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

Session 2:

$ RMU/BACKUP/LOG/ONLINE MF_PERSONNEL MF_PERSONNEL

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

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

```
Resource: nowait signal

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

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

10.2.5 Use of Oracle Rdb from Shareable Images

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

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

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

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

10.3 Oracle RMU Known Problems and Restrictions

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

10.3.1 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
```

10.3.2 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.

10.3.3 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.

10.3.4 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.

10.3.5 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.

10.3.6 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.

10.4 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.

10.4.1 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.

10.4.2 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.

10.4.3 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.

10.4.4 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.

10.4.5 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.

10.4.6 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.

10.4.7 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.

10.5 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.

10.5.1 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.)

10.5.2 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.

10.5.3 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 10−2 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

10.5.4 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.

10.5.5 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.

10.5.6 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