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

Table of Contents

Oracle® Rdb for OpenVMS

Release Notes

Release 7.2.2.0

January 2008

Oracle Rdb Release Notes, Release 7.2.2.0 for OpenVMS

Contents

Preface

Purpose of This Manual

This manual contains release notes for Oracle Rdb Release 7.2.2.0. 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 7.2.2.0.

Document Structure

This manual consists of the following chapters:

Chapter 1	Describes how to install Oracle Rdb Release 7.2.2.0.
Chapter 2	Describes problems corrected in Oracle Rdb Release 7.2.2.0.
Chapter 3	Describes problems corrected in Oracle Rdb Release 7.2.1.4.
Chapter 4	Describes problems corrected in Oracle Rdb Release 7.2.1.3.
Chapter 5	Describes problems corrected in Oracle Rdb Release 7.2.1.2.
Chapter 6	Describes problems corrected in Oracle Rdb Release 7.2.1.1.
Chapter 7	Describes problems corrected in Oracle Rdb Release 7.2.1.0.
Chapter 8	Describes enhancements introduced in Oracle Rdb Release 7.2.2.0.
Chapter 9	Describes enhancements introduced in Oracle Rdb Release 7.2.1.4.
Chapter 10	Provides information not currently available in the Oracle Rdb documentation set.
Chapter 11	Describes problems, restrictions, and workarounds known to exist in Oracle Rdb Release 7.2.2.0.

Chapter 1
Installing Oracle Rdb Release 7.2.2.0

This software update is installed using the OpenVMS VMSINSTAL utility.

NOTE

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

1.1 Oracle Rdb on HP OpenVMS Industry Standard 64

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

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

1.2 Requirements

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

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

1.3 Intel Itanium Processor 9100 "Montvale" Support

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

1.4 Maximum OpenVMS Version Check

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

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

1.5 Database Format Changed

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

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

1.6 Using Databases from Releases Earlier than V7.0

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

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

1.7 Invoking the VMSINSTAL Procedure

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

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

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

  ```
  @SYS$UPDATE:VMSINSTAL RDBV72200IM device−name
  ```
- To install the Oracle Rdb for OpenVMS Alpha kit that is compiled to run on all Alpha platforms:

  ```
  @SYS$UPDATE:VMSINSTAL RDBV72200AM device−name
  ```
- To install the Oracle Rdb for OpenVMS Alpha kit that is performance targeted for Alpha EV56 and later platforms:

  ```
  @SYS$UPDATE:VMSINSTAL RDBV72201AM device−name
  ```

device−name

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

1.8 Stopping the Installation

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

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

1.9 After Installing Oracle Rdb

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

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

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

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

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

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

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

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

1.10 VMS$MEM_RESIDENT_USER Rights Identifier Required

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

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

1.11 Installation, Configuration, Migration, Upgrade Suggestions

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

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

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

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

Table 1−1 Migration Suggestions

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

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

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

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

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

Chapter 2
Software Errors Fixed in Oracle Rdb Release 7.2.2.0

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.2.0.

2.1 Software Errors Fixed That Apply to All Interfaces

2.1.1 Potential System Crash Using VLM Global Buffers

In very rare conditions, when using the Global Buffers VLM (Very Large Memory) feature, it was possible for Oracle Rdb to cause an OpenVMS system crash with a bugcheck type of "MACHINECHK, Machine check while in kernel mode". The MACHINECHK can be triggered by an invalid PFN (page frame number) value being loaded into a PTE (page table entry).

The workaround to this problem is to discontinue use of the Oracle Rdb Global Buffers VLM (Very Large Memory) feature by specifying "ALTER DATABASE ... GLOBAL BUFFERS ...LARGE MEMORY IS DISABLED".

This problem has been corrected in Oracle Rdb Release 7.2.2. The Oracle Rdb VLM feature no longer can access past the end of the allocated PFN map.

2.1.2 Query Bugchecks After Upgrading From Release 7.1.1 to Release 7.1.5.1

Bug 6448202

A query bugchecks after a customer upgrades from Release 7.1.1 to Release 7.1.5.1. The query joins several tables with three other complicated views.

The query succeeds if the dynamic optimizer is disabled by defining the following SQL flags:

```
set flags 'max_stability';
```

This problem was introduced in Oracle Rdb Release 7.1.4.2 by a fix for Bug 4597186 but no problem has been encountered until the recent customer report. One of the retrieval blocks under a multiply nested cross strategy applied the dynamic optimizer LEAF node by walking up to the top parent query.

This problem has been corrected in Oracle Rdb Release 7.2.2.

2.1.3 Query Loops and Hangs Upgrading from Rdb V7.0 to V7.2

Bug 6459494

A query loops and hangs after the customer upgrades from Rdb Release 7.0.6 to an Rdb 7.2 release. The following query is a simplified one that displays the same behavior (very slow performance):

```
set flags 'strategy';
SELECT count(*)
     FROM PM P, KOSE K
```

```
     WHERE
         P.PROD = 'RSS' AND
         EXISTS(SELECT *
                 FROM KSU_V KV
                 WHERE
                     P.PROD = KV.PROD AND
                     P.PROD = K.PROD );
Aggregate
Cross block of 2 entries
  Cross block entry 1
    Match
      Outer loop      (zig-zag)
        Index only retrieval of relation KOSE
          Index name  IDX_KOSE [1:1]
      Inner loop      (zig-zag)
        Index only retrieval of relation PM
          Index name  IDX_PM [1:1]
  Cross block entry 2
    Conjunct        Aggregate-F1    Conjunct
    Merge of 2 entries
      Merge block entry 1
      Conjunct        Index only retrieval of relation KSU
        Index name  KSU_IDX [0:0]
      Merge block entry 2
      Cross block of 3 entries
        Cross block entry 1
          Conjunct
          Merge of 1 entries
            Merge block entry 1
            Reduce  Conjunct
            Index only retrieval of relation KSU
              Index name  KSU_IDX [1:1]
        Cross block entry 2
          Conjunct
          Merge of 1 entries
            Merge block entry 1
            Index only retrieval of relation KOSEIUNIT_PSIYO
              Index name  KSU_IDX_1 [4:4]
        Cross block entry 3
          Conjunct        Aggregate-F1
          Index only retrieval of relation KSU
            Index name  KSU_IDX [6:6]
```

The view KSU_V contains a UNION All clause.

```
create view KSU_V
    (PROD, ATTCD, UGRP, UBLK, USEQ, PSIYO) AS
    select C2.PROD, C2.ATTCD, C2.UGRP, C2.UBLK, C2.USEQ, C2.PSIYO
        from KSU C2
    union all
    select C3.F1, C3.F2, C3.F3, C3.F4, C3.F5, C6.PSIYO
        from
            (select C5.PROD, C5.ATTCD, C5.UGRP, C5.UBLK, C5.USEQ
                from KSU C5
                group by C5.PROD, C5.ATTCD, C5.UGRP, C5.UBLK, C5.USEQ)
            as C3 (F1, F2, F3, F4, F5),
            (select C7.PROD, C7.UGRP, C7.UBLK, C7.USEQ, C7.PSIYO
                from KOSEIUNIT_PSIYO C7
                where ((((C7.PROD = C3.F1)
                    and (C7.UGRP = C3.F3))
                    and (C7.UBLK = C3.F4))
                    and (C7.USEQ = C3.F5)))
```

```
        as C6 (PROD, UGRP, UBLK, USEQ, PSIYO)
    where (not exists (select * from KSU C8
        where ((((((C8.PROD = C3.F1)
            and (C8.ATTCD = C3.F2))
            and (C8.UGRP = C3.F3))
            and (C8.UBLK = C3.F4))
            and (C8.USEQ = C3.F5))
            and (C8.PSIYO = C6.PSIYO))));
```

The problem also exists in both Rdb Release 7.1.5.1 as well as Rdb Release 7.2.1.3. However, the query performs really fast on Rdb Release 7.0.9.

There is no workaround for this problem since this is a regression caused by the fixes for Bugs 5567495 and 4747999 in Rdb Release 7.1.5.1.

The presence of the first leg of the UNION in the view causes the strategy to use the full index scan of the following offending index.

```
  Cross block entry 2
    Conjunct        Aggregate-F1    Conjunct
    Merge of 2 entries
      Merge block entry 1
      Conjunct        Index only retrieval of relation KSU
        Index name  KSU_IDX [0:0]               <==
      Merge block entry 2
```

The index KSU_IDX is defined as:

```
create index KSU_IDX on KSYU
        (PROD, ATTCD, UGRP, UBLK, USEQ, PSIYO);
```

This problem has been corrected in Oracle Rdb Release 7.2.2.

2.1.4 Possible Bugcheck in RDMS$$INTERP_COMPILE

Bug 6505717

In several rare cases using the run−time native compiler on I64 systems, it is possible for the MOV_x instructions to trigger a bugcheck within RDMS$$INTERP_COMPILE with a footprint similar to the following:

```
***** Exception at FFFFFFFF8603B040 :
    RDMSHRP72\RDMS$$INTERP_COMPILE + 0004CC71
%COSI-F-BUGCHECK, internal consistency failure
Saved PC = FFFFFFFF85EFAE90 : RDMSHRP72\RDMS$$GEVX_RET_FINISH + 00000DD0
Saved PC = FFFFFFFF85E5A520 : RDMSHRP72\RDMS$$CREATE_EMOD + 00003CD0
Saved PC = FFFFFFFF85EDB9A0 : RDMSHRP72\RDMS$$SETUP_ACTION + 000000F0
Saved PC = FFFFFFFF85EBA850 : RDMSHRP72\RDMS$$PRE_EXECUTION + 00002020
Saved PC = FFFFFFFF85E874B0 : RDMSHRP72\RDMS$$COMPILE_FOR_IF + 00004340
Saved PC = FFFFFFFF85E79D00 : RDMSHRP72\RDMS$$COMPILE_STMT + 00001420
Saved PC = FFFFFFFF85E78D20 : RDMSHRP72\RDMS$$COMPILE_STMT + 00000440
Saved PC = FFFFFFFF85E7CC30 : RDMSHRP72\RDMS$$COMPILE_STMT + 00004350
Saved PC = FFFFFFFF85E78D20 : RDMSHRP72\RDMS$$COMPILE_STMT + 00000440
Saved PC = FFFFFFFF85E78D20 : RDMSHRP72\RDMS$$COMPILE_STMT + 00000440
Saved PC = FFFFFFFF85E767F0 : RDMSHRP72\RDMS$$DSDI_COMPILE + 00001F10
Saved PC = FFFFFFFF85FD3CC0 : RDMSHRP72\RDMS$TOP_COMPILE_REQUEST + 00002350
```

```
Saved PC = FFFFFFFF867261A0 : RDMSHRP72\AMAC$EMUL_CMPC5 + 00002040
Saved PC = FFFFFFFF86381C60 : RDMSHRP72\KODSTREAM$JACKET + 00000130
```

This problem has been corrected in Oracle Rdb Release 7.2.2. The code now correctly handles the possible source and destination addressing modes for all combinations.

2.1.5 Duplicate NOGDBL Name or RDMS−F−CANTCREGBL and COSI−F−BADPARAM When Attempting to Open Database With Shared Memory Mapped Resident

Bug 6495572

When using the database attribute SHARED MEMORY IS PROCESS RESIDENT, it was possible (perhaps only on a heavily loaded system with a very large global section) for the Rdb monitor process to detect an unexpected re−mapping of a global section. The monitor process would then re−attempt to create the database global section with a unique name but would errantly leave the prior global section mapped.

A user process then attempting to reopen or access a database could see fatal errors similar to the following:

```
%RDMS−F−CANTOPENDB, database could not be opened as requested
−RDMS−F−CANTCREGBL, error creating and mapping database global section
−COSI−F−BADPARAM, bad parameter value
```

The monitor log file may contain a "Duplicate NOGDBL name" message similar to the following:

```
6−OCT−2007 23:01:03.24 − Received open database request from 203589E8:0
  − process name _FTA170:, user RDB_ADMIN
  − database name "DSA58:[RDB.DATA]RESDB.RDB;1" [_DSA58] (13,1,0)
  − Duplicate NOGDBL name "RDM72NDSA58000D0001000000000000" detected ...
    retrying
  − opening as monitor ID 1
  − cluster watcher is active
  − sending normal open database reply to 203589E8:0
```

As a workaround, set the database to not use RESIDENT shared memory.

This problem has been corrected in Oracle Rdb Release 7.2.2. The Oracle Rdb monitor process unmaps the shared memory when it detects an unexpected re−mapping of a global section.

2.1.6 Possible Bugcheck With BADPAGNUM at PIOFETCH$WITHIN_DB After Drop and Recreate Logical Area On Cluster

Bug 6603641

In a multi−node cluster environment, it is possible when dropping and recreating logical areas for a bugcheck at PIOFETCH$WITHIN_DB to occur similar to the following "footprint".

```
RDMS−F−CANTREADDBS, error reading pages 148:10881−10881
RDMS−F−BADPAGNUM, page 10881 is out of valid range (1:2637)
```

```
for physical area 148
Exception occurred at RDMSHRP72\PIOFETCH$WITHIN_DB + 000005B8
Called from RDMSHRP72\PIOFETCH$FETCH + 000002E4
Called from RDMSHRP72\PIO$FETCH + 00000914
Called from RDMSHRP72\DIOLAREA$FIND_GOOD_PAGE + 00000214
Called from RDMSHRP72\DIO$RECORD_LOCATION + 0000021C
Called from RDMSHRP72\DIOSTORE$STORE_SEG + 000000E4
Called from RDMSHRP72\DIO$STORE + 000002A4
```

This problem could occur if a logical area was dropped on one node and then another logical area was added using the same logical area number but a different physical area. Another node in the cluster might have used a stale "next page" pointer for the logical area that was invalid for the new physical area.

This problem has been corrected in Oracle Rdb Release 7.2.2. Oracle Rdb now correctly validates that the "next page" pointer for the logical area is a valid possible page within the physical storage area.

2.1.7 Query Slows Down With Index Full Scan Upgrading to Rdb 7.2

Bug 6617515

A query slows down with index full scan after a customer upgrades from Rdb V7.0 to Rdb V7.2. The following query is a simplified one that still displays the same behavior.

```
set flags 'strategy';
select abc.GCD
from T0 t0,
      (select GCD
       from T1
       where GCD = (select GCD
                           from T2
                           where HBAN = t0.HBAN
                           and YSEQ = (select min(YSEQ)
                                        from T2
                                        where HBAN = t0.HBAN))) abc
  where YMD = '20070619';
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation T0
      Index name  IDX_T0 [1:1]
  Cross block entry 2
    Merge of 1 entries
      Merge block entry 1
      Cross block of 2 entries
        Cross block entry 1
          Aggregate
          Cross block of 2 entries
            Cross block entry 1
              Aggregate       Conjunct
              Index only retrieval of relation T2
                Index name  IDX_T2 [0:0]
            Cross block entry 2
              Index only retrieval of relation T2
                Index name  IDX_T2 [2:2]
        Cross block entry 2
          Index only retrieval of relation T1
            Index name  IDX_T1 [1:1]
```

The indices are defined as follows:

```
create index IDX_T0 on T0 (YMD, HBAN);
create index IDX_T1 on T1 (GCD)
create index IDX_T2 on T2 (HBAN, YSEQ, GCD);
```

The following is the strategy output from Rdb v7.0:

```
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation T0
      Index name  IDX_T0 [1:1]
  Cross block entry 2
    Merge of 1 entries
      Merge block entry 1
      Cross block of 2 entries
        Cross block entry 1
          Aggregate
          Cross block of 2 entries
            Cross block entry 1
              Aggregate       Index only retrieval of relation T2
                Index name  IDX_T2 [1:1]      Min key lookup
            Cross block entry 2
              Index only retrieval of relation T2
                Index name  IDX_T2 [2:2]
        Cross block entry 2
          Index only retrieval of relation T1
            Index name  IDX_T1 [1:1]
```

The only difference is the following lines:

```
From 7.2 output:
              Aggregate       Conjunct
              Index only retrieval of relation T2
                Index name  IDX_T2 [0:0]

From 7.0 output:
              Aggregate       Index only retrieval of relation T2
                Index name  IDX_T2 [1:1]      Min key lookup
```

The problem exists in Rdb V7.1 as well as Rdb V7.2 while the query performs quickly on Rdb Release 7.0.9.

There seems to be no workaround for this problem since this is a regression caused by the fixes for Bugs 5567495 and 4747999 in Rdb Release 7.2.1.0.

This problem has been corrected in Oracle Rdb Release 7.2.2.

2.1.8 Possible Query Slow Down Due to CONCAT Optimization

Bugs 6618080 and 6617696

In Oracle Rdb Release 7.2.1, the CONCAT operator was re−implemented to support more arguments and better concatenation performance. At the same time, the optimizer was enhanced to allow better index usage

by rewriting the CONCAT operation as a series of AND expressions. Unfortunately, the change may cause a switch from Match Join strategy to Cross Join strategy and an associated slow down of the query.

This problem has been corrected in Oracle Rdb Release 7.2.2. The CONCAT optimization has been corrected so that the optimizer can use it to implement a Match Join strategy.

2.1.9 Call Stack Not Symbolized In Bugcheck Dump Files With Resident Images On Alpha

Bug 6634194

In certain cases on Alpha systems with Rdb images installed /RESIDENT, routines are not correctly symbolized in the call stack of a bugcheck dump file. The following example shows one possible "footprint" of routines not being symbolized where they might otherwise be (note the system−space virtual addresses indicative of a resident image):

```
***** Exception at FFFFFFFF81653234 : Image RDMSHRP72 + 004DD234
%COSI−F−BUGCHECK, internal consistency failure
Saved PC = FFFFFFFF81640E88 : Image RDMSHRP72 + 004CAE88
Saved PC = FFFFFFFF816415D4 : Image RDMSHRP72 + 004CB5D4
Saved PC = FFFFFFFF81461464 : Image RDMSHRP72 + 002EB464
Saved PC = FFFFFFFF814401C4 : Image RDMSHRP72 + 002CA1C4
Saved PC = FFFFFFFF8143FB70 : Image RDMSHRP72 + 002C9B70
Saved PC = FFFFFFFF8143FB70 : Image RDMSHRP72 + 002C9B70
Saved PC = FFFFFFFF8144401C : Image RDMSHRP72 + 002CE01C
Saved PC = FFFFFFFF814440DC : Image RDMSHRP72 + 002CE0DC
Saved PC = FFFFFFFF8143FC78 : Image RDMSHRP72 + 002C9C78
Saved PC = FFFFFFFF8144401C : Image RDMSHRP72 + 002CE01C
Saved PC = FFFFFFFF814440DC : Image RDMSHRP72 + 002CE0DC
Saved PC = FFFFFFFF8143FC78 : Image RDMSHRP72 + 002C9C78
Saved PC = FFFFFFFF81443574 : Image RDMSHRP72 + 002CD574
Saved PC = FFFFFFFF81443750 : Image RDMSHRP72 + 002CD750
Saved PC = 00000000014B9DCC : symbol not found
Saved PC = FFFFFFFF8145A5F4 : Image RDMSHRP72 + 002E45F4
Saved PC = FFFFFFFF8122849C : Image RDMSHRP72 + 000B249C
Saved PC = FFFFFFFF81688294 : Image RDMSHRP72 + 00512294
```

As a workaround, do not install images /RESIDENT.

This problem has been corrected in Oracle Rdb Release 7.2.2. The call stack is correctly symbolized.

2.1.10 Bugcheck at RDMS__CS_TRANS_D

Bug 6633348

A database using the WIN_LATIN1 character set (as shown in the following example):

```
SQL> show character set;
Default character set is DEC_MCS
National character set is DEC_MCS
Identifier character set is DEC_MCS
Literal character set is DEC_MCS
Display character set is UNSPECIFIED
```

```
Alias RDB$DBHANDLE:
        Identifier character set is WIN_LATIN1
        Default character set is WIN_LATIN1
        National character set is UNICODE
SQL>
```

Could cause a bugcheck similar to:

```
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=
0000000000000188, PC=FFFFFFFF80002D40, PS=0000000B
Saved PC = 00000000803981C0 : RDMSHRP721\RDMS__CS_TRANS_D + 00000210
```

This was a problem on Itanium systems only.

This problem has been corrected in Oracle Rdb Release 7.2.2.

2.1.11 Possible Query Slow Down Due to the Fix for Bug 5192800

Bug 6619860

A fix for Bug 5192800 was made in Rdb Release 7.2.1.0 and causes the current customer's query to slow down significantly.

The query in Bug 5192800 involves an inner join and a left outer join operation with ON clause containing an equality predicate with a subselect query of MAX aggregate.

```
SELECT SM.ACCT_NUM, H.ACCT_NUM
  FROM
    SM SM INNER JOIN SC SC
    ON (SC.ACCT_NUM = SM.ACCT_NUM  AND
        EXISTS (SELECT * FROM EX EX
                 WHERE EX.ACCT_NUM = SC.ACCT_NUM))
    LEFT OUTER JOIN
    H H
    OH H.ACCT_NUM = SM.ACCT_NUM
    AND H.TIM_STMP =
            (SELECT MAC(H2.TIM_STMP)
              FROM H H2
              WHERE H2.ACCT_NUM = SM.ACCT_NUM );
```

The current customer's query does not involve an outer join operation but it contains two EXIST subselect queries.

```
SELECT  * FROM
     ( SELECT  HB FROM T1 WHERE  KBN = '1') TT1,
     ( SELECT  T2.HB
         FROM  T2,
               (SELECT  KBN FROM T3 WHERE   ZK_KBN ='1') TT3
         WHERE T2.KBN = TT3.KBN
           AND NOT EXISTS( SELECT *  FROM T2
                           WHERE KBN  =   '101'
                             AND ORDER_NO  <> ORD_NO )
           AND NOT EXISTS( SELECT *
                            FROM ( SELECT * FROM T2
                                    WHERE KBN =  '101'
```

```
                                    AND ORDER_NO  <> ORD_NO ) N1,
                                ( SELECT * FROM T2
                                    WHERE KBN = '201' ) N2
                            WHERE N1.ORD_NO = N2.ORD_NO
                              AND N1.HB     = N2.HB
                              AND N1.YMD    = N2.YMD )
        GROUP BY T2.HB
        HAVING T2.HB = TT1.HB
    )   T4;
```

This problem has been corrected in Oracle Rdb Release 7.2.2.

2.1.12 Bugcheck in DIOCCH$FETCH_SNAP_SEG

Bug 5240329

In unusual conditions when using the row cache feature, it was possible for access to an internal data structure to be incorrectly synchronized. This problem could cause a read−only transaction to fail with a bugcheck having a "footprint" similar to the following example (though other symptoms are possible):

```
***** Exception at 010BA554 : DIOCCH$FETCH_SNAP_SEG + 000009E4
%COSI−F−BUGCHECK, internal consistency failure
Saved PC = 010A8F14 : DIOFETCH$FETCH_ONE_LINE + 00000994
Saved PC = 010A95C8 : DIO$FETCH_DBKEY + 000002F8
```

This problem has been corrected in Oracle Rdb Release 7.2.2. The data structure access is now correctly synchronized between read−only and read−write transactions. All customers using the Row Cache feature are encouraged to upgrade.

2.2 SQL Errors Fixed

2.2.1 %SQL−W−LOOK_FOR_STT With NOT NULL Column Constraint

Bug 6157999

Whenever a NOT NULL constraint was specified for a column as the second or subsequent constraint, CREATE/ALTER TABLE would fail with a "%SQL−W−LOOK_FOR_STT".

For example, the following SQL statement would exhibit the problem:

```
SQL>create table tab1 (col1 char(10) unique not null);
create table tab1 (col1 char(10) unique not null);
                                          ^
%SQL−W−LOOK_FOR_STT, Syntax error, looking for:
%SQL−W−LOOK_FOR_CON,            DEFERRABLE,
%SQL−F−LOOK_FOR_FIN,    found NULL instead
```

The problem could be avoided by specifying the NOT NULL constraint prior to any other constraints. For example, the SQL statement above would succeed if written as follows:

```
create table tab1 (col1 char(10) not null unique);
```

This problem has been corrected in Oracle Rdb Release 7.2.2.

2.2.2 WAIT and NOWAIT Flags Not Used by Sequences

Bug 6487720

Prior versions of Oracle Rdb ignored the WAIT and NOWAIT flags defined for sequences. Therefore, all wait operations were controlled by the settings of the current transaction. When a transaction specified the NOWAIT clause, a reference to NEXTVAL might fail with a LOCK_CONFLICT error.

```
%RDB−E−LOCK_CONFLICT, request failed due to locked resource
−RDMS−F−LCKCNFLCT, lock conflict on client '........SEQ_....'
000000015F514553000000010000001900000000
```

This problem has been corrected in Oracle Rdb Release 7.2.2. WAIT and NOWAIT clauses of the ALTER SEQUENCE and CREATE SEQUENCE command now function as documented in the Oracle Rdb SQL Reference Manual.

2.2.3 Unexpected DEADLOCK Returned By CREATE TABLE

Bugs 6629124 and 5060366

In prior versions of Oracle Rdb, CREATE TABLE might, in rare cases, report a DEADLOCK error if the CREATE TABLE statement included COMPUTED BY, AUTOMATIC AS or DEFAULT clauses that referenced columns in the current table definitions.

The following example shows this unexpected error.

```
SQL> create table sample
cont>    (c_key char(16)
cont>    ,ckey_a automatic as upper (c_key)
cont>    );
%RDB-E-DEADLOCK, request failed due to resource deadlock
-RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-DEADLOCK, deadlock on client
'Table id 32 (SAMP)' 504D41530000002000000000400000055
```

This problem only occurs when the allocated RDB$RELATION_ID exceeds the maximum allowed (8192). In this case, Rdb attempts to reuse an old value previously assigned to a table and that has subsequently been freed by a DROP TABLE.

This problem has been corrected in Oracle Rdb Release 7.2.2. Rdb now correctly locates an unused relation id value without the deadlock error being reported.

2.2.4 Unexpected NOTGROFLD Error Reported for DECODE Function

Oracle Rdb allows expressions to be used in the GROUP BY clause. These expressions must match exactly expressions in the SELECT clause. In prior versions, SQL unexpectedly reported an NOTGROFLD error for the DECODE builtin as shown in the following example.

```
SQL> select ace_ord_type,
cont>       decode(ace_ord_status, 'CO', 1, 0),
cont>       count(*)
cont> from ace_orders
cont> group by ace_ord_type,
cont>          decode(ace_ord_status, 'CO', 1, 0);
%SQL-F-NOTGROFLD, Column ACE_ORD_STATUS cannot be referred to in the select
list, ORDER BY, or HAVING clause because it is not in the GROUP BY clause
```

As a workaround, the simple CASE expression can be used instead of the DECODE builtin function. For instance, the DECODE above could be converted to:

```
case ace_ord_status
    when 'CO' then 1
    else 0
end
```

This problem has been corrected in Oracle Rdb Release 7.2.2. DECODE is now correctly handled by SQL when performing semantic checks for GROUP BY expressions.

2.2.5 Memory Leak Corrected for the LIKE Operator

Bug 6405763

In prior versions of Oracle Rdb, there existed a small memory leak associated with the LIKE operator. After many interactions of queries using LIKE, the process could exceed the PGFLQUOTA for the process.

This problem has been corrected in Oracle Rdb Release 7.2.2.

2.3 RMU Errors Fixed

2.3.1 RMU/CONVERT Did Not Set the AIP Page Number Field in the Logical Area Entry

For Oracle Rdb RMU V7.2, a new field was added to the logical area entries in the Area Inventory Pages in Rdb databases. This field contained the AIP page number where the logical area entry was located. When an Rdb database of a version prior to V7.2 was converted to V7.2 using the /COMMIT qualifier (the default) or the /NOCOMMIT qualifier, the new field which contained this AIP page number was not initialized but contained a value of zero. This did not cause database corruption and the page number field would be set by subsequent operations which referenced the logical area such as an RMU/SET AIP/LENGTH=n/LAREA=n or RMU/SET AIP/REBUILD_SPAMS/LAREA=n. However, until it was initialized, the AIP page number field in the AIP entry would have a value of zero which would be displayed in a command like RMU/SHOW AIP/LAREA=n. This has been fixed and now RMU/CONVERT will properly set the AIP page number fields in the logical area entries in the AIP pages for an RMU/CONVERT to version V7.2.

The following example shows the problem that has been fixed. After converting the database to V7.2 an RMU/SHOW AIP command for a logical area would show an AIP page number of zero. Note that for a mixed format storage area, unlike a uniform format storage area, it is correct for the Area Bit Map page number field to be zero. This problem is only for the AIP page number field, not for the ABM page number field in the AIP entries which is handled correctly by RMU/CONVERT.

```
$ @SYS$LIBRARY:RDB$SETVER 7.2
$ RMU/CONVERT/COMMIT MF_PERSONNEL
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.2-00
%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
$ RMU/SHOW AIP/LAREA=80 MF_PERSONNEL

Logical area name EMPLOYEES
Type: TABLE
Logical area 80 in mixed physical area 5
Physical area name EMPIDS_OVER
Record length 126
AIP page number: 0
ABM page number: 0
Snapshot Enabled TSN: 60
```

A workaround for this problem is to do an operation on the logical area which will initialize the AIP page number entry for that logical area in the AIP pages of the database, such as an RMU/SET AIP command.

```
$ @SYS$LIBRARY:RDB$SETVER 7.2
$ RMU/CONVERT/COMMIT MF_PERSONNEL
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.2-00
%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
$ RMU/SHOW AIP/LAREA=80 MF_PERSONNEL

Logical area name EMPLOYEES
Type: TABLE
Logical area 80 in mixed physical area 5
```

```
Physical area name EMPIDS_OVER
Record length 126
AIP page number: 0
ABM page number: 0
Snapshot Enabled TSN: 60
$ RMU/SET AIP/LAREA=80/LENGTH=126 MF_PERSONNEL
$ RMU/SHOW AIP/LAREA=80 MF_PERSONNEL

Logical area name EMPLOYEES
Type: TABLE
Logical area 80 in mixed physical area 5
Physical area name EMPIDS_OVER
Record length 126
AIP page number: 150
ABM page number: 0
Snapshot Enabled TSN: 60
```

This problem has been corrected in Oracle Rdb Release 7.2.2.

2.3.2 ACCVIO When Using RMU/RESTORE With a Compressed Saveset

An RMU/RESTORE fails with an ACCVIO when starting the restore from the 2nd file of a compressed saveset. It is more likely to occur when used with encryption due to more random data found in the save set.

```
$ RMU/RESTORE/NOCDD LDA100:[000000]RBF1,LDA200: −
     /DISK=READER=1/NOLOG/ENCR=(VALUE=A2345678901234567890123456789012,
     ALG=AESCBC256)
%SYSTEM−F−ACCVIO, access violation, reason mask=00, virtual address=000002C3,
PC=0039EEF4, ...
%RMU−I−BUGCHKDMP, generating bugcheck dump file ...
```

This problem has been corrected in Oracle Rdb Release 7.2.2.

2.3.3 RMU Verify Ready of Read Only Storage Areas Could Return Error

Bug 6598519

An RMU/VERIFY Ready of an Oracle Rdb database storage area defined as READ ONLY could return an RMU−E−READONLY error unless /TRANSACTION_TYPE=READ_ONLY was specified. For RMU/VERIFY, /TRANSACTION_TYPE=WRITE is the default. This happened because, if the /TRANSACTION_TYPE was not READ ONLY, RMU attemped to ready the snapshot area associated with the READ ONLY storage area twice: first at the time the READ ONLY storage area was readied and then a second time, just before the snapshot area was to be verified. When RMU/VERIFY tried to ready the snapshot area associated with the READ ONLY area the second time, the RMU−E−READONLY error was returned.

This is now fixed and even if /TRANSACTION_TYPE=WRITE (the default), RMU/VERIFY will check to see if a storage area is defined as READ ONLY and ready it and the snapshot area associated with it in the correct way. Note that returning this error does not end the verify of the database, which continues with the next storage area. Also, when this error is returned, the READ ONLY storage area is verified but the snapshot area associated with the read only storage area is not verified.

The following example shows that if an Rdb database storage area was defined READ ONLY and RMU/VERIFY/TRANSACTION_TYPE=READ_ONLY was not specified, an RMU−E−READONLY error was returned.

```
$ SQL
ALTER DATABASE FILENAME MF_PERSONNEL
 ADD STORAGE AREA U_EMPIDS_MID PAGE FORMAT IS UNIFORM
ALTER DATABASE FILENAME MF_PERSONNEL ALTER STORAGE AREA U_EMPIDS_MID READ ONLY;
EXIT;
$ RMU/VERIFY/ALL/NOLOG MF_PERSONNEL
%RMU−E−BADREADY,  error readying storage area U_EMPIDS_MID
$ RMU/VERIFY/ALL/NOLOG/TRANSACTION_TYPE=WRITE MF_PERSONNEL
%RMU−E−BADREADY,  error readying storage area U_EMPIDS_MID
```

This problem can be avoided by specifying /TRANSACTION_TYPE=READ_ONLY for the verify.

```
$ RMU/VERIFY/ALL/NOLOG/TRANSACTION_TYPE=READ_ONLY MF_PERSONNEL
$
```

This problem has been corrected in Oracle Rdb Release 7.2.2.

2.3.4 RMU/MOVE/ROOT Did Not Retain the Unjournaled Changes Warning

Bug 2967642

The RMU/MOVE_AREA/ROOT command, which specifies that the database root should be moved as well as storage areas, incorrectly cleared the entry in the moved root which indicates that unjournaled changes have been made to an Oracle Rdb database storage area. If /ROOT was not specified with the RMU/MOVE_AREA command, this information was correctly maintained in the unmoved database root file. This problem has now been fixed and RMU/MOVE_AREA/ROOT will retain in the moved root file the information that some unjournaled changes have been made to the database that are not saved in an After Image Journal file.

The following example shows the problem that has now been fixed. RMU/MOVE_AREA without the /ROOT qualifier retained the unjournaled changes information in the unmoved Rdb database root file but RMU/MOVE_AREA/ROOT did not retain this information in the moved root file.

```
$ RMU/DUMP/OUTPUT=BEFORE.TXT TEST.RDB
$ search before.txt "Non−journalled database modifications have been made"
      − WARNING: Non−journalled database modifications have been made
$ RMU/MOVE_AREA/NOLOG/DIRECTORY=DISK:[DIRECTORY] TEST.RDB
%RMU−W−DOFULLBCK, full database backup should be done to ensure future recovery
$ RMU/DUMP/OUTPUT=AFTER.TXT TEST.RDB
$ search after.txt "Non−journalled database modifications have been made"
      − WARNING: Non−journalled database modifications have been made
$
$ RMU/DUMP/OUTPUT=BEFORE.TXT TEST.RDB
$ search before.txt "Non−journalled database modifications have been made"
      − WARNING: Non−journalled database modifications have been made
$ RMU/MOVE_AREA/NOLOG/DIRECTORY=DISK:[DIRECTORY]/ROOT=TEST_MOVED TEST.RDB
%RMU−I−AIJRSTAVL, 0 after−image journals available for use
%RMU−I−AIJISOFF, after−image journaling has been disabled
%RMU−W−DOFULLBCK, full database backup should be done to ensure future recovery
$ RMU/DUMP/OUTPUT=AFTER.TXT TEST_MOVED.RDB
$ search after.txt "Non−journalled database modifications have been made"
```

```
%SEARCH-I-NOMATCHES, no strings matched
```

This problem has been corrected in Oracle Rdb Release 7.2.2.

2.3.5 RMU /POPULATE_CACHE Bugchecks in PIOFETCH$WITHIN_DB

Bug 6655876

In rare cases, the RMU /POPULATE_CACHE command could bugcheck when attempting to read snapshot record copies. The following "footprint" will be seen in the bugcheck dump file:

```
Exception occurred at RMU72\PIOFETCH$WITHIN_DB + 000001E8
Called from RMU72\PIOFETCH$FETCH + 000002E4
Called from RMU72\PIO$FETCH + 00000914
Called from RMU72\DIOCCH$FETCH_SNAP_SEG + 000005E4
Called from RMU72\DIOFETCH$FETCH_ONE_LINE + 00000738
Called from RMU72\DIO$FETCH_DBKEY + 000002A4
Called from RMU72\PSIISCAN$WALK_BTREE_TOP_DOWN + 00000218
Called from RMU72\RMUPOP$FETCH_ONE_SORTED_INDEX + 000002EC
Called from RMU72\RMUPOP$FETCH_AREAS + 00000304
Called from RMU72\RMUCLI$POPULATE_CACHE + 00000414
```

This issue has been corrected in Oracle Rdb Release 7.2.2. RMU now correctly readies both the live and snapshot storage areas as required.

Chapter 3
Software Errors Fixed in Oracle Rdb Release 7.2.1.4

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.1.4.

3.1 Software Errors Fixed That Apply to All Interfaces

3.1.1 CPU Bound Loop in PIOUTL$SERVICE_PAGE_BLASTS

Bug 6111009

The ACMS server process gets stuck in a CPU bound loop. PC samples taken from the process or from a forced bugcheck dump point to locations in routine PIOUTL$SERVICE_PAGE_BLASTS.

The problem was a missed synchronization to prevent delivery of blocking ASTs. The outcome was the code inserted elements (BCBs) twice into a queue (BLASTED_QUEUE) causing a loop in the queue linkage.

The problem has been corrected in Oracle Rdb Release 7.2.1.4. A new flag (in the BCB) has been added together with interlocked test instructions to prevent this situation.

3.1.2 Wrong Result From Zigzag Match

Bug 6126475

A query with zigzag match strategy returns the wrong result (using the outline bug_match_outline which applies the index T1_SC_PC_NDX at the outer leg).

```
set flags 'strategy, detail, sort';

SELECT E.PCODE, P.PKEY
  FROM T1 E,
       T2 P
 WHERE E.SCODE = 'AXA'
   AND P.SNO = 4
   AND E.SCODE = P.SCODE
   AND E.PKEY = P.PKEY  ;
~S: Outline "BUG_MATCH_OUTLINE" used
~S#0003
Tables:
  0 = T1
  1 = T2
Conjunct: (0.SCODE = 1.SCODE) AND (0.PKEY = 1.PKEY)
Match
  Outer loop
    Sort: 0.SCODE(a), 0.PKEY(a)
      SortId# 4., # Keys 4
        Item# 1, Dtype: 2, Order: 0, Off: 0, Len: 1
        Item# 2, Dtype: 14, Order: 0, Off: 1, Len: 3
        Item# 3, Dtype: 2, Order: 0, Off: 4, Len: 1
        Item# 4, Dtype: 8, Order: 0, Off: 5, Len: 4
        LRL: 32, NoDups:0, Blks:6, EqlKey:0, WkFls: 2
    Leaf#01 BgrOnly 0:T1 Card=3
      Bool: 0.SCODE = 'AXA'
      BgrNdx1 T1_SC_PC_NDX [1:1] Fan=16
        Keys: 0.SCODE = 'AXA'
```

```
  Inner loop      (zig-zag)
    Index only retrieval of relation 1:T2
      Index name  T2_SC_SN_PK_NDX [2:2]
        Keys: (1.SCODE = 'AXA') AND (1.SNO = 4)
0 rows selected
```

The query works if the outline is changed to use the index T1_SC_PK_PC_NDX instead of T1_SC_PC_NDX.

```
create outline BUG_MATCH_OUTLINE
id '045E2C384D284FD1061D1BF58CF8872D'
mode 0
as (
  query (
-- For loop
    subquery (
      T1 0         access path index
--    T1_SC_PC_NDX        -- replace it with T1_SC_PK_PC_NDX
      T1_SC_PK_PC_NDX
        join by match to
      T2 1      access path index
      T2_SC_SN_PK_NDX
      )
    )
  )
compliance optional     ;

SELECT E.PCODE, P.PKEY
  FROM T1 E,
       T2 P
 WHERE E.SCODE = 'AXA'
   AND P.SNO = 4
   AND E.SCODE = P.SCODE
   AND E.PKEY = P.PKEY  ;
~S: Outline "BUG_MATCH_OUTLINE" used
Tables:
  0 = T1
  1 = T2
Conjunct: (0.SCODE = 1.SCODE) AND (0.PKEY = 1.PKEY)
Match
  Outer loop      (zig-zag)
    Index only retrieval of relation 0:T1
      Index name  T1_SC_PK_PC_NDX [1:1]
        Keys: 0.SCODE = 'AXA'
  Inner loop      (zig-zag)
    Index only retrieval of relation 1:T2
      Index name  T2_SC_SN_PK_NDX [2:2]
        Keys: (1.SCODE = 'AXA') AND (1.SNO = 4)
 E.PCODE        P.PKEY
     850         84900
     850         84910
2 rows selected
```

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

1. The main query joins two tables using a one−sided zigzag match strategy where the zigzag skip occurs only at the outer leg.
2. One of the match join keys (i.e. PKEY) is NOT included in the outer index and thus a sort is applied to produce the required index order at the outer leg.

This problem has been corrected in Oracle Rdb Release 7.2.1.4.

3.1.3 Wrong Result From Outer Join With MISSING Predicate

Bug 6404754

Wrong results were reported from an outer join query with a "Missing" predicate. The customer's query is simplified into the following reproducer:

```
create table TAB1 (COL1 CHAR(4));
create table TAB2 (COL1 CHAR(4));

insert into TAB1 values ('1030');
insert into TAB1 values ('1420');

insert into TAB2 values ('1030');
insert into TAB2 values ('1420');

commit;

! the following query should return 0 rows but returns 2 rows incorrectly:
!
SELECT *
  FROM TAB1 AAA,
 (SELECT TAB1.COL1 FROM TAB1, TAB2 WHERE TAB1.COL1 = TAB2.COL1(+) ) BBB
   WHERE AAA.COL1 = BBB.COL1(+) AND BBB.COL1 IS NULL;
Tables:
  0 = TAB1
  1 = TAB1
  2 = TAB2
Conjunct: MISSING (1.COL1)
Cross block of 2 entries        (Left Outer Join)
  Cross block entry 1
    Get     Retrieval sequentially of relation 0:TAB1
  Cross block entry 2
    Conjunct: MISSING (1.COL1)         <== See NOTE below
    Conjunct: 0.COL1 = 1.COL1
    Merge of 1 entries
      Merge block entry 1
      Conjunct: MISSING (1.COL1)        <== See NOTE below
      Cross block of 2 entries        (Left Outer Join)
        Cross block entry 1
          Get     Retrieval sequentially of relation 1:TAB1
        Cross block entry 2
          Conjunct: 1.COL1 = 2.COL1
          Get     Retrieval sequentially of relation 2:TAB2
 AAA.COL1   BBB.COL1
 1030       NULL
 1420       NULL
2 rows selected
```

NOTE:: The conjunct is incorrectly pushed down from the main query across the left outer join query and thus causing the wrong result.

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

1. The main query is a left outer join query between a table and a derived table.
2. The derived table is derived from another left outer join query of two tables.
3. The WHERE clause contains an IS NULL predicate referencing the join column of the derived table.

This problem has been corrected in Oracle Rdb Release 7.2.1.4.

3.2 SQL Errors Fixed

3.2.1 Unexpected Column Reordering After an ALTER TABLE ... ADD COLUMN Statement

Bug 6148536

In prior releases of Oracle Rdb, it was possible for SHOW TABLE on tables that were often altered to show that the columns had been unexpectedly reordered. This unusual case happens when the RDB$FIELD_ID values exceed 65535. Oracle discourages such tables in production as the very large RDB$FIELD_ID causes the null bit vector of these rows to be quite large. The many ALTER TABLE operations will also propagate description rows in the RDB$FIELD_VERSIONS table that will affect metadata loading by the Rdb Server.

The large RDB$FIELD_ID value will force the row to be followed by a null bit vector large enough to hold all fields. If compression is disabled, then this will lead to fragmented rows and excessive I/O during table processing. Therefore, the correct solution for this problem is to unload the table data and recreate the table. A SQL EXPORT and IMPORT would also have the same result.

The cause of the unexpected reordering was an assumption that the RDB$FIELD_ID would remain 16 bits in size and so the high order 16 bits of RDB$FIELD_POSITION were used to encode the AFTER COLUMN and BEFORE COLUMN clauses for ALTER TABLE. The overflow was incorrectly interpreted as a column reordering command.

This problem has been corrected in Oracle Rdb Release 7.2.1.4. Oracle Rdb now manages the reordering in a different way that no longer uses the RDB$FIELD_ID column values.

3.2.2 SET ALL CONSTRAINTS Causes SQLCODE −1005 With Compound Statement

Bugs 462035 and 1403770

If a Precompiled SQL or SQL Module Language program executed a statement with no active transaction while constraints were being evaluated immediately, the statement would fail with a SQLCODE of −1005 and associated Rdb error message "%RDB−E−BAD_TRANS_HANDL". Typically, the SQL statement might be a compound statement or a call to a stored or external procedure since these statements do not require a transaction to be active.

For example, the following Precompiled SQL program commits its transaction and then immediately executes a SQL compound statement. That compound statement would fail with a SQLCODE of −1005.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
EXEC SQL DECLARE ALIAS FILENAME PERSONNEL;
EXEC SQL INCLUDE SQLCA;
main()
{
    EXEC SQL set transaction;
```

```
    printf("SET TRANS SQLCODE is %ld\n", SQLCA.SQLCODE);
    EXEC SQL set all constraints on;
    printf("SET ALL CONSTR ON SQLCODE is %ld\n", SQLCA.SQLCODE);
    EXEC SQL commit;
    printf("COMMIT SQLCODE is %ld\n", SQLCA.SQLCODE);
    EXEC SQL begin set transaction; end;
    printf("compound SQLCODE is %ld\n", SQLCA.SQLCODE);
}
```

This problem has been corrected in Oracle Rdb Release 7.2.1.4.

3.2.3 GET DIAGNOSTICS in Distributed Transaction Makes Transaction Non−distributed

Bug 972038

Under some circumstances, a compound statement containing a GET DIAGNOSTICS statement would cause a Precompiled SQL or SQL Module Language program built with "TRANSACTION_DEFAULT=DISTRIBUTED" to start a non−distributed transaction. The compound statement also needed a statement that requires a transaction context, such as an INSERT. In such a case, if there was no distributed transaction active, the SQL$PRE or SQL$MOD program would not start one prior to executing the compound statement. This would cause the server to implicitly start a non−distributed transaction for the database attach.

For example, if the following Precompiled C code were program compiled with a SQLOPTION of "TRANSACTION_DEFAULT=DISTRIBUTED", the condition could occur.

```
...
EXEC SQL DECLARE ALIAS FILENAME PERSONNEL;
...
main ()
{
    long rc;
    long status;
    short iosb[4];
    long flag = 2;
    long tid[4];
...
    status = sys$start_transw(
        0, /* efn */
        flag, /* flags */
        iosb, /* iosb */
        0, /* astadr */
        0, /* astprm */
        tid /* tid */
    );
...
    EXEC SQL BEGIN
        update employees set last_name = 'Toliver'
            where employee_id = '00164';
        get diagnostics :rc = row_count;
    END;
    printf("row count = %d\n", rc);
...
    status = sys$end_transw(
        0, /* efn */
        0, /* flag */
```

```
        iosb, /* iosb */
        0, /* astadr */
        0, /* astprm */
        tid /* tid */
    );
...
}
```

In the above example, the presence of the "GET DIAGNOSTICS" statement in the compound statement would cause the client program to not automatically join the distributed transaction initiated by the call to SYS$START_TRANSW(). Since there would then be no Rdb transaction in progress for the PERSONNEL database, the Rdb server would implicitly start a one−phase transaction independent from the application's distributed transaction. The call to SYS$END_TRANSW() would not then commit the Rdb transaction because it was not in a branch participating in the distributed transaction.

SQL$PRE and SQL$MOD have been modified so that when "TRANSACTION_DEFAULT=DISTRIBUTED" is specified, code is generated which does implicitly join any existing distributed transaction prior to executing a compound statement containing a "GET DIAGNOSTICS" statement.

The problem could be worked around by making sure a distributed transaction was active when such a compound statement was executed.

This problem has been corrected in Oracle Rdb Release 7.2.1.4.

3.2.4 IMPORT DATABASE Command Losing Some Command Line Database Attributes

Bug 6311200

In prior releases of Oracle Rdb, the SQL IMPORT DATABASE command would not correctly inherit the SHARED MEMORY IS PROCESS RESIDENT clause or the LARGE MEMORY IS ENABLED attribute of the GLOBAL BUFFERS ARE ENABLED clause. The corresponding attributes were always inherited from the interchange file (.RBR).

This has been corrected in Oracle Rdb Release 7.2.1.4. However, a workaround is to use an ALTER DATABASE statement immediately following the IMPORT DATABASE command to set these attributes. The RMU/EXTRACT/ITEM=DATABASE command can be used to verify these settings.

3.2.5 SET FLAGS 'TRANSACTION' Now Displays Global Transaction ID

Bug 3107989

When the flag TRANSACTION is defined during the start of a distributed (aka two phase commit) transaction, Oracle Rdb now displays the global transaction id associated with this database transaction. This allows the collection of statistics related to distributed transactions.

A line starting with "~T Global transaction id:" is displayed immediately following the "~T Start_transaction" log message as shown in this example.

```
$ DEFINE RDMS$SET_FLAGS TRANSACTION
$ SQL$
SQL> ATTACH 'ALIAS A FILENAME DB$:MF_PERSONNEL';
SQL> ATTACH 'ALIAS B FILENAME DB$:PERSONNEL';
SQL> START TRANSACTION READ WRITE;
~T Compile transaction (1) on db: 1
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_WRITE (read write)
~T Start_transaction (1) on db: 1, db count=2
~T Global transaction id: 34C08373-41E3-11DC-ABB9-0008021A53CE
~T Compile transaction (1) on db: 2
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_WRITE (read write)
~T Start_transaction (1) on db: 2, db count=2
~T Global transaction id: 34C08373-41E3-11DC-ABB9-0008021A53CE
SQL>
```

If the transactions are not distributed, then this information is not displayed. Using the TEST_SYSTEM flag will disable this information.

This problem has been corrected in Oracle Rdb Release 7.2.1.4.

3.2.6 Unexpected Bugcheck From IMPORT DATABASE Command

Bug 6313120

Starting with Oracle Rdb Release 7.1.4, new system routines are created that reflect the mapping defined by the CREATE STORAGE MAP statement. These special routines are automatically generated by CREATE STORAGE MAP and should not be created manually. The SQL EXPORT DATABASE command for V7.1.4 and later versions will omit these specially marked storage mapping functions from the interchange file (.rbr).

However, if an older version of SQL is used to create the interchange file, such as in the case of multiversion V7.0 being installed on the same system, then the resulting interchange file will include copies of these routines.

In prior releases, SQL IMPORT DATABASE would bugcheck when processing such interchange files. These dumps would have a footprint similar to this example:

- Alpha OpenVMS 8.2
- Oracle Rdb Server 7.1.5.0.0
- Got a RDSBUGCHK.DMP
- SYSTEM-F-ACCVIO, access violation, virtual address=0000000000000000
- Exception occurred at COSI_MEM_FREE_VMLIST + 00000094
- Called from RDMS$$RELEASE_DDL_VM + 00000118
- Called from RDMS$$RELEASE_DDL_VM_HNDLR + 00000084
- Running image SQL$71.EXE
- Dump created: 31-JUL-2007 02:14:22.71
- Database root: 1DGA124:[TESTING]MF_PERSONNEL

This problem has been corrected in Oracle Rdb Release 7.2.1.4. SQL IMPORT DATABASE will now generate a message similar to this when attempting to import the duplicate routines.

```
%SQL-F-NOMODRES, unable to import module RDB$STORAGE_MAPS
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-RTNEXTS, there is another routine named EMPLOYEE_HISTORY in
this database
```

3.3 RDO and RDML Errors Fixed

3.3.1 %COBOL−F−AMBIGSYM, Ambiguous Reference With RDBPRE and COBOL

Bug 486335

Under certain circumstances, compiling an RDBPRE/COBOL program would result in generated code with duplicate names in a generated record definition. When the resulting COBOL code was submitted to the COBOL compiler, the following error message would be generated at the point the field with the duplicate name was used:

```
%COBOL−F−AMBIGSYM, Ambiguous reference − check name qualification ...
```

The record structures with the duplicate names were used by RDBPRE to store intermediate results during a cross−database update. The problem would manifest itself when columns in the source database had different types and/or sizes than the corresponding target columns in the other database. For example, consider a pair of Rdb databases defined by the following SQL statements.

```
create database filename essai1
create table a_table (
    a_zone1     SMALLINT,
    a_zone2     SMALLINT,
    a_zone3     INTEGER (2));
disconnect all;
create database filename essai
create table b_table (
    b_zone1     SMALLINT,
    b_zone2     INTEGER (3),
    b_zone3     SMALLINT (1));
disconnect all;
```

The generated code from the following RDBPRE/COBOL program would contain a record definition with a duplicate name.

```
IDENTIFICATION DIVISION.
PROGRAM−ID. tst.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING−STORAGE SECTION.
*
&RDB& INVOKE DATABASE b  = FILENAME "essai"
&RDB& INVOKE DATABASE a  = FILENAME "essai1"
*
PROCEDURE DIVISION.
DEB−PROG.
&RDB&    START−TRANSACTION ON B  USING (READ−WRITE
&RDB&                      RESERVING b.b_table            FOR SHARED READ)
&RDB&                    AND ON A  USING (READ−WRITE
&RDB&                      RESERVING a.a_table            FOR SHARED WRITE)
&RDB&    FOR   EC  IN b.b_table
&RDB&           WITH  EC.b_zone1  = 0
&RDB&           STORE ECC IN a.a_table USING
&RDB&                   ECC.a_zone2  = EC.b_zone2;
```

```
&RDB&                   ECC.a_zone3  = EC.b_zone3;
&RDB&           END_STORE
&RDB&    END_FOR
&RDB&    COMMIT
          stop run.
```

The problem could often be worked around by rearranging the column order in the STORE statement.

This problem has been corrected in Oracle Rdb Release 7.2.1.4.

3.3.2 %RDB−F−ARITH_EXCEPT With RDBPRE/COBOL and Literal in Expression

Bug 560785

In some cases, a STORE or MODIFY that involved an arithmetic expression with a literal in it would cause RDBPRE/COBOL to generate bad code. Specifically, the generated code would have a type mismatch between a BLR message and the corresponding items in the generated COBOL record for that message. At run time, this would often lead to error messages such as: "%RDB−F−ARITH_EXCEPT" and/or "−COSI−F−INTOVF".

The following example of a program that demonstrates the problem uses a PERSONNEL database with the JOBS relation altered so that the MAXIMUM_SALARY and MINIMUM_SALARY fields have a type of "signed word scale 0". The JOBS.RCO program from SQL$SAMPLES is modified as described below.

- The fields WS−SAL−MIN and WS−SAL−MAX are modified to be four byte integers as follows:

  ```
  01 WS-SAL-MIN                   PIC S9(4) COMP.
  01 WS-SAL-MAX                   PIC S9(4) COMP.
  ```
- The STORE−JOBS subroutine is modified to set the minimum and maximum salary variables to "1" and use an arithmetic expression involving a literal in the RDO statement for storing the jobs records as follows:

  ```
  STORE-JOBS.
          MOVE 1 TO WS-SAL-MIN.
          MOVE 1 TO WS-SAL-MAX.
  &RDB&   STORE J IN JOBS
  &RDB&      USING
  &RDB&           J.JOB_CODE              = J-CODE;
  &RDB&           J.WAGE_CLASS            = W-CLASS;
  &RDB&           J.JOB_TITLE             = J-TITLE;
  &RDB&           J.MINIMUM_SALARY        = WS-SAL-MIN + 1;
  &RDB&           J.MAXIMUM_SALARY        = WS-SAL-MAX + 2;
  &RDB&    END_STORE
  ```

When the modified JOBS.RCO program was compiled and linked, the resulting code would have a mismatch between the minimum and maximum salary fields in the generated COBOL record versus the message in the BLR. This caused run time errors.

This problem has been corrected in Oracle Rdb Release 7.2.1.4.

3.4 RMU Errors Fixed

3.4.1 RMU/RESTORE or /DUMP/BACKUP Bugchecks

Bug 6001187

If RMU detects a corrupted saveset, it prompts an interactive user to either 'QUIT or CONTINUE'. In some cases, the command exits with an ACCVIO before writing out the prompt message or it ACCVIOs if the user puts in a response other than 'QUIT' or 'CONTINUE'.

Example 1:

```
$ RMU/DUMP/BACK MFP.RBF
%RMU-I-DMPTXT_163, No dump option selected. Performing read check.
%RMU-E-READERR, error reading USER:[USERDIR]MFP.RBF;
-RMU-E-BLOCKCRC, software block CRC error
%RMU-E-INVBLKSIZ, invalid block size in backup file
%RMU-E-INVRECSIZ, invalid record size in backup file
%RMU-E-READERRS, excessive error rate reading USER:[USERDIR]MFP.RBF;
-RMU-E-BLOCKCRC, software block CRC error
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=...
```

Example 2:

```
$ RMU/DUMP/BACK MFP.RBF
%RMU-I-DMPTXT_163, No dump option selected. Performing read check.
%RMU-E-READERR, error reading USER:[USERDIR]MFP.RBF;
-RMU-E-BLOCKCRC, software block CRC error
%RMU-E-INVBLKSIZ, invalid block size in backup file
%RMU-E-INVRECSIZ, invalid record size in backup file
%RMU-E-READERRS, excessive error rate reading USER:[USERDIR]MFP.RBF;
-RMU-E-BLOCKCRC, software block CRC error
%RMU-I-SPECIFYC, specify option (QUIT or CONTINUE)
RMU> exit
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=...
```

If RMU detects a corrupted saveset on an IA64 system, it bugchecks after the user replies with 'QUIT'. RMU then loops writing out error messages.

Example 3:

```
$ RMU/RESTORE MFP.RBF
%RMU-E-READERRS, excessive error rate reading
IVMS_USER1:[FROEHLIN.TEST4]MFP.RBF;
-RMU-E-BLOCKCRC, software block CRC error
%RMU-I-SPECIFYC, specify option (QUIT or CONTINUE)
RMU> quit
%COSI-F-BUGCHECK, internal consistency failure
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-I-BUGCHKDMP, generating bugcheck dump file
USER:[USERDIR]RMUBUGCHK.DMP;
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at ...
%COSI-F-BUGCHECK, internal consistency failure
...loops with the following lines
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
```

```
%COSI-F-BUGCHECK, internal consistency failure
```

One reason for this problem was an invalid prompt string descriptor. The looping on IA64 was caused by an inappropriate condition handler.

These problems have been corrected in Oracle Rdb Release 7.2.1.4.

3.4.2 RMU/CONVERT Could Not Delete System Metadata Below the V7.0 Version

Bug 6270091

There was a restriction in V7.2 RMU/CONVERT where obsolete system metadata below the V7.0 version could not be deleted. Instead of deleting obsolete metadata for versions V6.1 and below, the conversion would be terminated, the database would be corrupt, and the following error would be output:

```
$   RMU/CONVERT/COMMIT MF_PERSONNEL
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.2-00
%RMU-I-LOGCONVRT, database root converted to current structure version
%RMU-E-CANTCVRT, cannot convert this database version
%RMU-F-FTL_CNV, Fatal error for CONVERT operation at  7-AUG-2007 22:14:50.50
```

This problem has been fixed and now obsolete metadata of version V6.1 and below will be deleted and the conversion will terminate normally. Note that problems like this demonstrate the importance of backing up any database before conversion.

Obsolete alternate metadata can be left in an Oracle Rdb database if RMU/CONVERT/NOCOMMIT or RMU/RESTORE/NOCOMMIT is specified to convert the database to a new metadata version to allow for testing the conversion but the Convert is not commited before going to the next version by specifying RMU/CONVERT/COMMIT or RMU/RESTORE/COMMIT. Note that COMMIT is the default. Specifying /NOCOMMIT retains two copies of the metadata in the database, the new current version and the previous alternate version. Once testing with the new current metadata version is completed, the user can specify RMU/CONVERT/COMMIT or RMU/RESTORE/COMMIT to delete the old obsolete alternate metadata and retain just the new current metadata, or if the user has problems he can do an RMU/CONVERT/ROLLBACK to delete the new current metadata version and restore the previous alternate metadata version as the current metadata version. Of course, doing a /ROLLBACK means the previous version of Rdb will have to be used to access the database.

If the database conversion is not commited before going to the next version, an obsolete alternate version of metadata will be retained in the database which will use unneeded logical area space in the database. In the case of this problem, which has been fixed, it also caused the convert to fail if obsolete alternate metadata of versions V6.1 and below had to be deleted.

What the CONVERT code does when going to a new version, even in the /NOCOMMIT case, is make a first pass to delete any existing alternate metadata because it is going to have to save the previous version of metadata as the new alternate metadata since only the current and the previous metadata versions can be saved by current database design. For example, let us start with a V6.0 version database that contains only a current 6.0 version of metadata − current/alternate versions of 6.0/0. If I do an RMU/CONVERT/NOCOMMIT to 7.0, I have 7.0/6.0 versions of metadata in the database. But when I do an RMU/CONVERT/NOCOMMIT to 7.1, I first delete the old alternate 6.0 metadata and at the end have 7.1/7.0 versions of metadata retained in the database. One example of the problem fixed by this bug would be a V6.1 database with current/alternate 6.1/0

versions of metadata. An RMU/CONVERT/NOCOMMIT to V7.1 creates 7.1/6.1 versions of metadata in the databse. An RMU/CONVERT/NOCOMMIT to V7.2 creates 7.2/7.1 versions of metadata, but first has to delete the obsolete 6.1 version of metadata. An RMU/CONVERT/COMMIT to V72 creates 7.2/0 versions of metadata. In both cases, since the obsolete 6.1 version of metadata has to be deleted, because RMU/CONVERT could not handle metadata versions V6.1 and below, the conversion would fail.

Note that now, for metadata versions of V6.1 and V6.0, all the metadata and the logical areas for the metadata will be deleted. However, for metadata versions of V5.1 and below, the pointer to the obsolete metadata will be deleted but the actual obsolete metadata and the logical areas containing the metadata will not be deleted. However, the database will be valid and the conversion will complete normally. Note also that the current and alternate metadata mentioned here does not include user created tables and indexes, but refers to the system tables and indexes created by Oracle Rdb which define the internal Rdb database structure.

The following example shows the problem that has been fixed. A database that starts out with current/alternate metadata versions of 6.0/0, when converted /NOCOMMIT to Rdb V7.1 has current/alternate metadata versions of 7.1/6.0. When converted /COMMIT (the default) to Rdb V7.2, the conversion failed since RMU/CONVERT was not able to delete obsolete versions of the metadata below V7.0.

```
$   @SYS$LIBRARY:RDB$SETVER 6.0
$   RMU/RESTORE/NOLOG/NOCDD/DIRECTORY=DEVICE:[DIRECTORY] -
 DEVICE:[DIRECTORY.60]MF_PERSONNEL.RBF
 %RMU-I-AIJRSTAVL, 0 after-image journals available for use
 %RMU-I-AIJISOFF, after-image journaling has been disabled
 %RMU-W-USERECCOM, Use the RMU/RECOVER command. The journals are not
 available.
$   @SYS$LIBRARY:RDB$SETVER 7.1
$   RMU/CONVERT/NOCOMMIT MF_PERSONNEL
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.1-00
%RMU-I-LOGCONVRT, database root converted to current structure version
%RMU-S-CVTDBSUC, database DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
 successfully converted from version V6.0 to V7.1
$   @SYS$LIBRARY:RDB$SETVER 7.2
$   RMU/CONVERT/COMMIT MF_PERSONNEL
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.2-00
%RMU-I-LOGCONVRT, database root converted to current structure version
%RMU-E-CANTCVRT, cannot convert this database version
%RMU-F-FTL_CNV, Fatal error for CONVERT operation at  7-AUG-2007
22:14:49.68
```

A workaround for this problem is to do an RMU/CONVERT/COMMIT of the database at the current Rdb version before converting the database to the next Rdb version.

This problem has been corrected in Oracle Rdb Release 7.2.1.4.

3.4.3 RMU /MOVE_AREA or /COPY_DATABASE With /BLOCKS_PER_PAGE May Fail

In certain cases, the RMU /MOVE_AREA or RMU /COPY_DATABASE command may fail when using the /BLOCKS_PER_PAGE qualifier. The actual symptoms of the failure include unexpected bugchecks. The cause of the problem was related to code errantly writing past the end of an allocated data structure.

As a workaround, do not use the /BLOCKS_PER_PAGE qualifier.

This problem has been corrected in Oracle Rdb Release 7.2.1.4.

3.4.4 RMU/COLLECT/INDEXES Gave No Error If An Index Did Not Exist

Bug 6316729

If RMU/COLLECT OPTIMIZER_STATISTICS /INDEX=(A,B,C) or RMU/SHOW OPTIMIZER_STATISTICS /INDEX=(A,B,C) was given the name of an index that did not exist in an Oracle Rdb database, it ignored the index but continued and returned no error.

This problem has been fixed and now if the user specifies a named index that does not exist in the database, the RMU/COLLECT OPTIMIZER or RMU/SHOW OPTIMIZER command will be aborted and an error will be returned.

```
$ rmu/collect opt mf_personnel /notables /log /index=bad_name
Start loading tables... at 28-AUG-2007 16:16:14.54
Done loading tables.... at 28-AUG-2007 16:16:14.54
Start loading indexes... at 28-AUG-2007 16:16:14.54
%RMU-F-INDNOTFND, index BAD_NAME does not exist in this database
%RMU-F-FTL_COL_STAT, Fatal error for COLLECT OPTIMIZER_STATISTICS operation at
28-AUG-2007 16:16:14.54
```

The following example shows the problem that has been fixed. Even though the named index "bad_name" does not exist, RMU/COLLECT OPTIMIZER_STATISTICS gives no error but continues and ignores the index.

```
$ rmu/collect opt mf_personnel /notables /log /index=bad_name
Start loading tables... at 28-AUG-2007 16:14:21.04
Done loading tables.... at 28-AUG-2007 16:14:21.05
Start loading indexes... at 28-AUG-2007 16:14:21.05
Done loading indexes.... at 28-AUG-2007 16:14:21.06
Start collecting btree index stats... at 28-AUG-2007 16:14:21.12
Done collecting btree index stats.... at 28-AUG-2007 16:14:21.12
Start collecting table & hash index stats... at 28-AUG-2007 16:14:21.12
Done collecting table & hash index stats.... at 28-AUG-2007 16:14:21.12
Start collecting workload stats... at 28-AUG-2007 16:14:21.17
Maximum memory required (bytes) = 0
Done collecting workload stats.... at 28-AUG-2007 16:14:21.18
Start calculating stats... at 28-AUG-2007 16:14:21.18
Done calculating stats.... at 28-AUG-2007 16:14:21.18
Start writing stats... at 28-AUG-2007 16:14:21.21
Done writing stats.... at 28-AUG-2007 16:14:21.21
```

This problem has been corrected in Oracle Rdb Release 7.2.1.4.

3.4.5 RMU OPTIMIZER_STATISTICS Commands Gave Wrong Fatal Error Exit Message

RMU/COLLECT OPTIMIZER_STATISTICS, RMU/DELETE OPTIMIZER_STATISTICS and RMU/INSERT OPTIMIZER_STATISTICS returned a fatal error exit message which did not name correctly the operation being performed. Note that the error was handled correctly but the fatal message put out before Oracle Rdb RMU exited was incorrect.

This problem has been fixed and now a fatal error exit message appropriate to the specific command operation will be output.

```
$ rmu/collect optimizer_statistics baddb
%RMU-W-BADDBNAME, can't find database root
DEVICE:[DIRECTORY]BADDB.RDB;
-RMS-E-FNF, file not found
%RMU-F-CANTOPNROO, cannot open root file "BADDB"
%RMU-F-FTL_COL_STAT, Fatal error for COLLECT OPTIMIZER_STATISTICS operation at
 29-AUG-2007 15:44:07.54
$ rmu/delete optimizer_statistics baddb
%RMU-W-BADDBNAME, can't find database root
DEVICE:[DIRECTORY]BADDB.RDB;
-RMS-E-FNF, file not found
%RMU-F-CANTOPNROO, cannot open root file "BADDB"
%RMU-F-FTL_DEL_STAT, Fatal error for DELETE OPTIMIZER_STATISTICS operation at
 29-AUG-2007 15:44:30.18
$ rmu/insert optimizer_statistics baddb
%RMU-W-BADDBNAME, can't find database root
DEVICE:[DIRECTORY]BADDB.RDB;
-RMS-E-FNF, file not found
%RMU-F-CANTOPNROO, cannot open root file "BADDB"
%RMU-F-FTL_INS_STAT, Fatal error for INSERT OPTIMIZER_STATISTICS operation at
 29-AUG-2007 15:44:50.48
```

The following example shows the problem that has been fixed. The fatal error exit message %RMU−F−FTL_ANA was returned by RMU. It did not correctly describe the command operation being performed.

```
$ rmu/collect optimizer_statistics baddb
%RMU-W-BADDBNAME, can't find database root
DEVICE:[DIRECTORY]BADDB.RDB;
-RMS-E-FNF, file not found
%RMU-F-CANTOPNROO, cannot open root file "BADDB"
%RMU-F-FTL_ANA, Fatal error for ANALYZE operation at 29-AUG-2007
 15:33:52.70
$ rmu/delete optimizer_statistics baddb
%RMU-W-BADDBNAME, can't find database root
DEVICE:[DIRECTORY]BADDB.RDB;
-RMS-E-FNF, file not found
%RMU-F-CANTOPNROO, cannot open root file "BADDB"
%RMU-F-FTL_ANA, Fatal error for ANALYZE operation at 29-AUG-2007
 15:34:05.65
$ rmu/insert optimizer_statistics baddb
%RMU-W-BADDBNAME, can't find database root
DEVICE:[DIRECTORY]BADDB.RDB;
-RMS-E-FNF, file not found
%RMU-F-CANTOPNROO, cannot open root file "BADDB"
%RMU-F-FTL_ANA, Fatal error for ANALYZE operation at 29-AUG-2007
 15:34:19.03
```

This problem has been corrected in Oracle Rdb Release 7.2.1.4.

3.4.6 RMU/CONVERT Created System Logical Areas of an Unspecified Type

Bug 6390145

Oracle Rdb RMU/CONVERT, when converting an Rdb database to the current version, did not specify the logical area type when creating logical areas for system tables and system indexes. This gave the logical areas containing system tables and indexes an "UNKNOWN" type when the database was converted. This did not corrupt the database but caused a command such as RMU/SHOW AIP to display "UNKNOWN" for the logical area type. Now the type for logical areas containing system tables and indexes created by RMU/CONVERT will correctly be "TABLE" and "SORTED INDEX".

This problem has been fixed as the following example shows. The correct type has been specified for the new system logical areas created by RMU/CONVERT.

```
$ rmu/convert/noconfirm mf_personnel
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.2-nn
%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
%RMU-I-CVTCOMSUC, CONVERT committed for
 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 to version V7.2
$ rmu/show aip/brief mf_personnel

*-------------------------------------------------------------------------------

* Logical Area Name             LArea PArea   Len Type

*-------------------------------------------------------------------------------

RDB$PROFILES                      114     1   140 TABLE
RDB$GRANTED_PROFILES              115     1    28 TABLE
RDB$TYPES                         116     1   148 TABLE
RDB$TYPE_FIELDS                   117     1   148 TABLE
RDB$WORKLOAD                      118     1    75 TABLE
RDB$TRIGGER_ACTIONS               119     1    84 TABLE
RDB$NDX_NDX_NAME_NDX              120     1   219 SORTED INDEX
RDB$NDX_REL_NAME_NDX              121     1   219 SORTED INDEX
RDB$NDX_SEG_NAM_FLD_POS_NDX       122     1   219 SORTED INDEX
RDB$REL_REL_NAME_NDX              123     1   219 SORTED INDEX
RDB$VER_REL_ID_VER_NDX            124     1   219 SORTED INDEX
```

The following example shows the problem that has been fixed. No type was specified for the new system logical areas created by RMU/CONVERT.

```
$ rmu/convert/noconfirm mf_personnel
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.2-nn
%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
%RMU-I-CVTCOMSUC, CONVERT committed for
 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 to version V7.2
$ rmu/show aip/brief mf_personnel

*-------------------------------------------------------------------------------

* Logical Area Name             LArea PArea   Len Type

*-------------------------------------------------------------------------------

RDB$PROFILES                      114     1   140 UNKNOWN
RDB$GRANTED_PROFILES              115     1    28 UNKNOWN
RDB$TYPES                         116     1   148 UNKNOWN
RDB$TYPE_FIELDS                   117     1   148 UNKNOWN
```

```
RDB$WORKLOAD                     118     1    75 UNKNOWN
RDB$TRIGGER_ACTIONS              119     1    84 UNKNOWN
RDB$NDX_NDX_NAME_NDX             120     1   219 UNKNOWN
RDB$NDX_REL_NAME_NDX             121     1   219 UNKNOWN
RDB$NDX_SEG_NAM_FLD_POS_NDX      122     1   219 UNKNOWN
RDB$REL_REL_NAME_NDX             123     1   219 UNKNOWN
RDB$VER_REL_ID_VER_NDX           124     1   219 UNKNOWN
```

This problem has been corrected in Oracle Rdb Release 7.2.1.4.

3.4.7 RMU/VERIFY/LOG VMS Exit Status Could Be Incorrect

Big 6315971

If RMU/VERIFY/LOG was specified and no errors or warnings occurred when an Oracle Rdb database was verified, the VMS exit status would be incorrectly set to a log message with Informational severity instead of a success status. This did not occur if /LOG was not specified. This problem has been fixed and now a success status will be output if /LOG is specified as it already is if /LOG is not specified.

The following example shows the problem that has been fixed. A successful RMU/VERIFY of an Rdb database would show a successful VMS exit status if /LOG was not specified but if /LOG was specified the exit status would be that of an informational log message. Now in both cases a successful VMS exit status will be set.

```
$ rmu/verify/all mf_personnel
$ show symbol $status
  $STATUS == "%X10000001"
$ rmu/verify/all/log mf_personnel
 %RMU-I-BGNROOVER, beginning root verification
 %RMU-I-ENDROOVER, completed root verification
 %RMU-I-BGNVCONST, beginning verification of constraints for database
  DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
 %RMU-I-ENDVCONST, completed verification of constraints for database
  DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1

 %RMU-S-ENDVERIFY, elapsed time for verification :    0 00:00:02.22
$ show symbol $status
  $STATUS == "%X12C8B51B"
```

This problem has been corrected in Oracle Rdb Release 7.2.1.4.

3.5 LogMiner Errors Fixed

3.5.1 RMU/UNLOAD/AFTER_JOURNAL Sort Work File Size Potentially Reduced

The RMU/UNLOAD/AFTER_JOURNAL performs a sort operation to eliminate duplicate record modifications for each transaction being extracted. In some cases (especially when record compression within the database was used), on−disk sort temporary work file allocation could become excessive because RMU was overestimating the maximum record length to be sorted.

This problem has been corrected in Oracle Rdb Release 7.2.1.4. The maximum record size to be sorted is now more accurately calculated. This may serve to reduce the size of the sort work files and avoid some work file IO.

3.5.2 RMU/UNLOAD/AFTER_JOURNAL /QUICK_SORT_LIMIT Qualifier

The RMU/UNLOAD/AFTER_JOURNAL performs a sort operation to eliminate duplicate record modifications for each transaction being extracted. For smaller sort cardinalities, an internal in−memory "quick sort" algorithm is used, otherwise the SORT32 algorithm is used. Previously, the limit for using the quick sort routine was a fixed value of 5000 records.

This restriction of a fixed value for the threshold has been relaxed in Oracle Rdb Release 7.2.1.4. A new qualifier /QUICK_SORT_LIMIT=n has been provided to allow explicitly controlling the maximum number of records that will be sorted with the in−memory algorithm. The default value is 5000. The minimum value is 10 and the maximum value is 100,000.

Larger values specified for the /QUICK_SORT_LIMIT qualifier may reduce sort work file IO at the expense of additional CPU time and/or memory consumption. A too small value may result in additional disk file IO. Oracle believes that, in general, the default value should be accepted.

3.6 RMU Show Statistics Errors Fixed

3.6.1 File Name Overwritten in File IO Statistics Display

Bug 6319121

Starting with Oracle Rdb Release 7.2.1, the name of the file is not displayed on the "File IO Statistics" screen of the RMU /SHOW STATISTICS utility. This problem was caused by a change in the statistics heading lines being shifted too far.

This problem has been corrected in Oracle Rdb Release 7.2.1.4. The file name is no longer overwritten.

3.6.2 Bugcheck When Moving Backwards To Device Information Display

Bug 6319159

Starting with Oracle Rdb Release 7.2.1, when moving backwards (using a "Page Up" key) into the "Device Information" screen, the RMU /SHOW STATISTICS utility may bugcheck. This problem was caused by an internal data structure being incorrectly configured while moving backwards (moving forwards through the screens did not cause this problem).

This problem has been corrected in Oracle Rdb Release 7.2.1.4.

3.6.3 RMU/SHOW STATISTICS /INPUT= Possible Errors or Bugchecks

Bug 6339722

Starting with Oracle Rdb Release 7.2.1, using the RMU /SHOW STATISTICS utility with an input binary file could result in errors while reading the input file. In addition, the "Elapsed:" time display field on line two of the screen heading was often left blank.

As a possible workaround, the /NOLOGICAL_AREA qualifier can be used on the RMU commands that read and write the binary file to avoid the problems.

These problems have been corrected in Oracle Rdb Release 7.2.1.4. The binary input file is correctly processed.

Chapter 4
Software Errors Fixed in Oracle Rdb Release 7.2.1.3

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.1.3.

4.1 Software Errors Fixed That Apply to All Interfaces

4.1.1 Possible Shared Memory Corruption When Multiple Databases Attached

Bug 5841667

Starting with Oracle Rdb Release 7.1.4 and Oracle Rdb Release 7.2, it was possible for shared memory to become corrupt. The corruption often would appear as (or would be caused by) data from one Rdb root file being written into the shared memory for another database. Once this corruption has occurred, reliability and functionality of the database and database users can be compromised.

Conditions leading to this corruption include:

- Processes accessing multiple databases
- Multiple database users
- Databases accessed from multiple nodes in a cluster
- Databases configured with "NUMBER OF CLUSTER NODES" ... "MULTIPLE INSTANCE"

The memory corruption was caused by incorrect I/O buffer synchronization while refreshing root file information into shared memory.

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

Note

Oracle strongly recommends that customers with applications or procedures that may attach to more than one database at a time upgrade to Oracle Rdb Release 7.2.1.3 or later to avoid potential memory corruption problems.

4.1.2 Remote TCP/IP Bugchecks If Database Is Configured For Internal Security Checking

Bug 6152325

When attaching remotely, using TCP/IP protocol, to a database that is configured for internal security checking, a bugcheck and a bugcheck dump was generated.

```
***** Exception at 0000000000A7730C : RDMSHRP72\RDMS$$DML$READY + 0000005C
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=00000000000001D4, PC=0000000000A7730C, PS=00000009
```

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.1.3 Incorrect Results From Complex Query Involving RDB$DATABASE System Table

Bug 6195672

In prior releases of Oracle Rdb V7.2, complex queries against the RDB$DATABASE system table could return insufficient rows. This is related to an optimization for the queries on this special table so that they always perform DBKEY retrieval.

The following example shows the problem. This query should return 4 rows from the CROSS join of the two row derived tables.

```
SQL> SELECT F.FLAG, A.ACCOUNT
cont> FROM
cont>   (SELECT 'Y' AS FLAG FROM RDB$DATABASE
cont>    UNION ALL
cont>    SELECT 'N' AS  FLAG FROM RDB$DATABASE) F,
cont>   (SELECT 'A' AS ACCOUNT FROM RDB$DATABASE
cont>    UNION ALL            ! note:: the query works if UNION
cont>    SELECT 'M' AS ACCOUNT FROM RDB$DATABASE) A ;
 F.FLAG   A.ACCOUNT
 N        A
 N        M
2 rows selected
SQL>
```

The workaround is to replace RDB$DATABASE in the query with another single row table.

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.1.4 ALTER DATABASE ... SHARED MEMORY IS PROCESS Did Not Disable RESIDENT

Previously it was not possible, while using SQL, to disable the RESIDENT attribute for database shared memory once it had been enabled.

In the following example, note that once enabled, the RESIDENT attribute does not get cleared when using the ALTER statement.

```
$ SQL$ CREATE DATA FILE FOO SHARED MEMORY IS PROCESS RESIDENT;
$ RMU/DUMP/HEAD/OUT=X.X FOO
$ SEARCH X.X MAPPED
    Database will be mapped in process space
      Shared memory will be mapped resident (OpenVMS Alpha only)
$ SQL$ ALTER DATA FILE FOO SHARED MEMORY IS PROCESS;
$ RMU/DUMP/HEAD/OUT=X.X FOO
$ SEARCH X.X MAPPED
    Database will be mapped in process space
      Shared memory will be mapped resident (OpenVMS Alpha only)
```

As a workaround, the "RMU/SET SHARED_MEMORY/TYPE=" command can be used to switch between SYSTEM, PROCESS and RESIDENT.

This problem has been corrected in Oracle Rdb Release 7.2.1.3. Using "ALTER DATABASE ... SHARED MEMORY IS PROCESS" now correctly clears the setting of the RESIDENT attribute.

4.1.5 Incomplete Error Handling for COSI−E−WORK_DEV

Bug 4898352

In the event that an Rdb sort work file is not a local, random access file, Rdb reported an incomplete error message. In the case that the database in question was remote, the remote RDBSERVER would fail with an access violation. For example, if the error were in an interactive SQL session for a local database, the incomplete error message would appear as follows:

```
%COSI−E−WORK_DEV, work file !AS must be on random access local device
```

In the above example, the "!AS" should have been replaced by the name of the problem sort work file, for example "NL:[]SORTWORK.TMP;".

If the error occurred on a remote database, the RDBSERVER would fail and the local session would see the following error messages:

```
%RDB−F−IO_ERROR, input or output error
−SYSTEM−F−LINKABORT, network partner aborted logical link
```

In the remote case, the NETSERVER.LOG for that session on the remote system would record a "%SYSTEM−F−ACCVIO" error.

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.1.6 Wrong Result From Query With NOT NULL Test

Bug 6041167

A wrong result may be reported by a query with a NOT NULL test, as in the following example.

```
SHOW INDEX T1_NDX;
Indexes on table T1:
T1_NDX                          with column COL_DATA
                                and column KEY_ID
  Duplicates are allowed
  Type is Sorted Ranked
    Duplicates are Compressed Bitmaps
  Key suffix compression is DISABLED
  Node size  430

SEL COL_DATA,KEY_ID,NUMBER FROM T1;
Get     Retrieval by index of relation T1
  Index name  T1_NDX [0:0]
    COL_DATA        KEY_ID        NUMBER
          55          NULL      96507257
          66          NULL      96509951
          68          NULL      96508028
3 rows selected
SELECT COUNT(*) FROM T1 WHERE KEY_ID IS NOT NULL;
Aggregate       Conjunct        Index only retrieval of relation T1
```

```
  Index name  T1_NDX [0:0]       Index counts lookup

           3
1 row selected
```

The query works if the column of the NOT NULL predicate is defined as the leading segment, as in the following example.

```
DROP INDEX T1_NDX;
CREATE INDEX T1_NDX  ON T1 (KEY_ID,COL_DATA)
    TYPE IS SORTED RANKED;
SELECT COUNT(*) FROM T1 WHERE KEY_ID IS NOT NULL;
Aggregate Index only retrieval of relation T1
  Index name  T1_NDX [0:0]   Index counts lookup
 0
1 row selected
```

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

1. The query selects the total count where the column is NOT null.
2. The column is defined as the trailing segment of the SORTED RANKED index.
3. The "Index counts lookup" was the chosen strategy.

A workaround for this problem is to disable the "Index counts lookup" for any affected queries. This is achieved using SET FLAGS 'NOCOUNT_SCAN'.

This problem affects the following Oracle Rdb releases: V7.1.4.4, V7.1.4.5, V7.1.5, V7.2.0.1, V7.2.0.2, V7.2.1, V7.2.1.1 and V7.2.1.2. This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.1.7 Bugcheck at COSI$TIMER_GET_REQIDT With RDMS−F−NOREQIDT

Bug 6069358

Applications using the fast commit feature that periodically detach and reattach to a database within the same program may eventually run out of Rdb timer blocks and bugcheck with a footprint similar to the following.

```
***** Exception at 01235994 : COSI$TIMER_GET_REQIDT + 00000294
%RDMS−F−NOREQIDT, reached internal maximum number of simultaneous
timer requests
Saved PC = 01235C38 : COSI_TIMER_SET + 00000288
Saved PC = 01235DA8 : COSI_TIMER_SLEEP + 00000078
Saved PC = 012A7548 : KOD$COMMIT + 00000508
Saved PC = 01079148 : RDMS$$INT_COMMIT_TRANSACTION + 00000328
Saved PC = 01078D14 : RDMS$TOP_COMMIT_TRANSACTION + 000001E4
```

The following example script fragment demonstrates this failure using a fast commit checkpoint interval of 1 second and a delay between transactions of just over two seconds (to allow enough time for the checkpoint timer to fire twice):

```
    .
    .
    .
$ DEFINE SQL$DATABASE DKA0:[DB]DB.RDB
```

```
$ DEFINE RDM$BIND_CKPT_TIME 1
    .
    .
    .
$ SQL$
    INSERT INTO C1 VALUES (1);
    COMMIT;
    $ WAIT 0:0:2.02
    DISCONNECT ALL;
        .
        . Repeat the sequence 101 times
        .
    INSERT INTO C1 VALUES (1);
    COMMIT;
    $ WAIT 0:0:2.02
    DISCONNECT ALL;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file
DISK$USER:[USER]RDSBUGCHK.DMP;
```

This problem has been corrected in Oracle Rdb Release 7.2.1.3. The problem was caused by an internal timer data structure being allocated but not being deallocated if the timer had expired. If the timer had not expired, the internal timer data structure was correctly deallocated. Thus, in some cases, the timer data structure was being "leaked" which could eventually lead to the bugcheck exception of "RDMS-F-NOREQIDT, reached internal maximum number of simultaneous timer requests".

4.1.8 Inappropriate Message When Dropping Journal if No Journals Exist

Bug 5059712

When a database has no journals defined, dropping a non-existant journal returns the somewhat misleading message "RDMS-F-AIJDISABLED, after-image journaling must be enabled for this operation" as in the following example:

```
$ SQL$
        CREATE DATABASE FILE AIJTST RESERVE 3 JOURNALS
        CREATE STORAGE AREA RDB$SYSTEM;
        DISCONNECT ALL;
        ALTER DATABASE FILENAME AIJTST DROP JOURNAL FOO;
%RDMS-F-AIJDISABLED, after-image journaling must be enabled for this operation
        EXIT;
$ EXIT
```

This minor issue has been corrected in Oracle Rdb Release 7.2.1.3. Rdb now returns the more accurate and explicit message "RDMS-F-NOSUCHAIJ, no such AIJ journal "journal-name"" as in this example:

```
$ SQL$
        CREATE DATABASE FILE AIJTST
        RESERVE 3 JOURNALS
        CREATE STORAGE AREA RDB$SYSTEM;
        DISCONNECT ALL;
        ALTER DATABASE FILENAME AIJTST DROP JOURNAL FOO;
%RDMS-F-NOSUCHAIJ, no such AIJ journal "FOO"
        EXIT;
$ EXIT
```

4.1.9 Bugcheck From UNION Query With Constant Boolean

Bug 6125013

It was reported that a bugcheck was received from a UNION query with a constant boolean, as shown in the following example.

```
SELECT A_CLEARER_ID
FROM (SELECT A_DATE,A_CLEARER_ID
       FROM   T_CLEARED_BY
       UNION ALL
       SELECT A_DATE,A_CLEARER_ID
       FROM   T_CLEARED_BY
) AS   TMP
WHERE  A_DATE = DATE'2007-06-15' AND 1=1;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file DISK:[DIRECTORY]RDSBUGCHK.DMP;
```

The query works if the constant predicate "1=1" is removed:

```
SELECT A_CLEARER_ID
FROM (SELECT A_DATE,A_CLEARER_ID
       FROM   T_CLEARED_BY
       UNION ALL
       SELECT A_DATE,A_CLEARER_ID
       FROM   T_CLEARED_BY
) AS   TMP
WHERE  A_DATE = DATE'2007-06-15';
Merge of 1 entries
  Merge block entry 1
  Merge of 2 entries
    Merge block entry 1
    Conjunct        Index only retrieval of relation T_CLEARED_BY
      Index name  U1_CLEARED_BY [1:1]
    Merge block entry 2
    Conjunct        Index only retrieval of relation T_CLEARED_BY
      Index name  U1_CLEARED_BY [1:1]
0 rows selected
```

This problem occurs when the main query selects from a derived table of a union query with a boolean predicate such as "1=1" in the WHERE clause.

This problem affects the following Oracle Rdb versions: V7.1.4.4, V7.1.4.5, V7.1.5, V7.1.5.1, V7.2.0.1, V7.2.0.2 and V7.2.1, V7.2.1.1 and V7.2.1.2. This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.2 SQL Errors Fixed

4.2.1 Unexpected Constraint Activation After ALTER TABLE ... DISABLE CONSTRAINT

In prior releases of Oracle Rdb, the ALTER TABLE command did not enable or disable constraints already active in the current session. Constraints are enabled and disabled using ALTER TABLE ... ENABLE CONSTRAINT constraint−name, ALTER TABLE ... DISABLE CONSTRAINT constraint−name, ALTER TABLE ... ENABLE ALL CONSTRAINTS and ALTER TABLE ... DISABLE ALL CONSTRAINTS.

The following example shows that the constraint is still checked during the session even though it was disabled using ALTER TABLE.

```
SQL> create table my_con3
cont>     (c1 char
cont>     ,c2 char
cont>     ,constraint MYC1 check (c1 in ('a','b')) deferrable);
SQL> commit;
SQL>
SQL> insert into my_con3 (c1,c2) values ('d','e');
1 row inserted
SQL> commit;
%RDB−E−INTEG_FAIL, violation of constraint MYC1 caused operation to fail
−RDB−F−ON_DB, on database RDB$DEFAULT_CONNECTION
SQL> rollback;
SQL>
SQL> insert into my_con3 (c1,c2) values ('a','b');
1 row inserted
SQL> commit;
SQL> rollback;
SQL>
SQL> alter table my_con3 disable constraint myc1;
SQL> show table (constraint) my_con3;
Information for table MY_CON3

Table constraints for MY_CON3:
MYC1
 Check constraint
 Table constraint for MY_CON3
 Evaluated on COMMIT
 Source:
        CHECK (c1 in ('a','b'))
Constraint is disabled

Constraints referencing table MY_CON3:
 No Constraints found

SQL> commit;
SQL>
SQL> insert into my_con3 (c1,c2) values ('d','e');
1 row inserted
SQL> commit;
%RDB−E−INTEG_FAIL, violation of constraint MYC1 caused operation to fail
−RDB−F−ON_DB, on database RDB$DEFAULT_CONNECTION
```

This problem has been corrected in Oracle Rdb Release 7.2.1.3. Oracle Rdb now correctly applies the enable/disable to the active constraints in the session.

4.2.2 Bugcheck with "SQL$BLRXPR − 15" on Cross−DB Insert

Bug 4307036

In certain cases, an INSERT statement with a cross−database select would fail and produce a SQLBUGCHK.DMP that contains the following line:

```
%SQL−F−BUGCHK, There has been a fatal error. Please contact your Oracle
support representative. SQL$BLRXPR − 15
```

Such INSERT statements would involve a DISTINCT keyword or a CASE expression in the SELECT. For example, consider the following database and variable declarations:

```
$ SQL$
SQL> create database filename temp1;
SQL> create table ctab (
cont> destination char(4),
cont> state       char(2)
cont> );
SQL> insert into ctab values ('ABCD','ME');
1 row inserted
SQL> insert into ctab values ('EFGH','ME');
1 row inserted
SQL> insert into ctab values ('IJKL','MI');
1 row inserted
SQL> insert into ctab values ('ABCD','ME');
1 row inserted
SQL> commit;
SQL> disconnect all;
SQL> create database filename temp2;
SQL> create table tb1 (
cont> ccode   char(4),
cont> snum    integer
cont> );
SQL> commit;
SQL> disconnect all;
SQL> att 'alias db2 file temp2';
SQL> att 'alias db1 file temp1';
SQL> declare : run_id integer = 5;
SQL> declare :xx char(2) = 'xx';
SQL> declare :suffix char(2) = 'ME';
SQL> declare :hv2 integer = 1;
```

In the above context, the following cross−database INSERT statements would fail as shown:

```
SQL> insert into db2.tb1
cont>  (select distinct destination, :run_id from db1.ctab where state='ME');
%RDMS−I−BUGCHKDMP, generating bugcheck dump file MY$DISK:[MY_HOME]SQLBUGCHK.DMP;
%SQL−F−BUGCHK, There has been a fatal error. Please contact your Oracle support
representative. SQL$BLRXPR − 15
SQL> insert into db2.tb1
cont> (select destination,
cont>  case when :suffix indicator :hv2 is null
```

```
cont>       then 1 else 2 end case
cont>  from db1.ctab);
%RDMS-I-BUGCHKDMP, generating bugcheck dump file MY$DISK:[MY_HOME]SQLBUGCHK.DMP;
%SQL-F-BUGCHK, There has been a fatal error. Please contact your Oracle support
representative. SQL$BLRXPR - 15
```

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.2.3 Comments Not Always Recognized by SQL$PRE/COBOL

Bug 4751103

The SQL Precompiler was not properly recognizing COBOL comment statements. Most of the time this makes no difference but in certain cases, such as in the middle of a record definition, it was possible for names to be missed resulting in a error such as a %SQL-F-HVNOTDECL.

For example, consider the following COBOL program with embedded SQL.

```
000000 IDENTIFICATION DIVISION.
000000 PROGRAM-ID.    DUMMYIO.
000000 AUTHOR.        RDB ENGINEERING.
000000
000000 ENVIRONMENT DIVISION.
000000
000000 CONFIGURATION SECTION.
000000 SOURCE-COMPUTER. ALPHA-RDB.
000000 OBJECT-COMPUTER. ALPHA-RDB.
000000
000000 DATA DIVISION.
000000
000000 WORKING-STORAGE SECTION.
000000
000000     EXEC SQL INCLUDE SQLCA END-EXEC
000000
000000 01  HOST-DATA.
000000     02  HOST-CODE                  PIC  X(03).
000000* COMPUTATIONAL ITEMS
000000     02  OLD-TOTAL-COST             PIC S9(09)V99    COMP.
000000     02  OLD-TOTAL-TAX              PIC S9(09)V99    COMP.
000000* END OF COMPUTATIONAL ITEMS
000000     02  HOST-REASON                PIC  X(60).
000000     02  USER-CODE                  PIC  X(05).
000000
000000 01  SELECTION-RANGES.
000000     02  NO-RANGE                   PIC  X(1).
000000
000000 PROCEDURE DIVISION.
000000
000000 THE-PROGRAM.
000000
000000     PERFORM SELECT-HOST-RECORD.
000000     MOVE SPACE TO NO-RANGE.
000000
000000 RETURN-TO-CALLER.
000000     EXIT PROGRAM.
000000
000000 SELECT-HOST-RECORD.
```

```
000000     EXEC SQL SELECT COUNT(*)
000000     INTO :OLD-TOTAL-COST
000000     FROM RDB$DATABASE
000000     END-EXEC.
```

In the above program, there are two comments inside the definition of the HOST−DATA record. The SQL Precompiler would not successfully parse these comments and, as a result, would give the following error message:

```
000000     INTO :OLD-TOTAL-COST
                1
%SQL-F-HVNOTDECL, (1) Host variable OLD-TOTAL-COST was not declared
```

In the bug report, the embedded comments were a pair of alignment compiler directives which are special COBOL comment lines that appear similar to the following:

```
000000 01  HOST-DATA.
000000     02  HOST-CODE                  PIC  X(03).
000000*DC SET ALIGNMENT
000000     02  OLD-TOTAL-COST             PIC S9(09)V99    COMP.
000000     02  OLD-TOTAL-TAX              PIC S9(09)V99    COMP.
000000*DC END-SET ALIGNMENT
000000     02  HOST-REASON                PIC  X(60).
000000     02  USER-CODE                  PIC  X(05).
```

The problem can be worked around by removing the comments from the middle of the record definition. If the comments are alignment compiler directives, such as the ALIGNMENT directives shown above or the PADALIGN directive, these can be repositioned to bracket the entire record and the same COBOL code will be generated. For example, the record definition shown above can be recoded as:

```
000000*DC SET ALIGNMENT
000000 01  HOST-DATA.
000000     02  HOST-CODE                  PIC  X(03).
000000     02  OLD-TOTAL-COST             PIC S9(09)V99    COMP.
000000     02  OLD-TOTAL-TAX              PIC S9(09)V99    COMP.
000000     02  HOST-REASON                PIC  X(60).
000000     02  USER-CODE                  PIC  X(05).
000000*DC END-SET ALIGNMENT
```

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.2.4 Incorrect Data Displayed from CONCAT Operator

Bugs 6002006 and 6034530

In Oracle Rdb V7.2.1, V7.2.1.1 and V7.2.1.2, the CONCAT operator (also ||) may generate incorrect results. The following example shows the error as ".." instead of the expected "−>".

```
SQL> set flags 'trace';
SQL>
SQL> begin
cont>  for :a as each row of
cont>           select s_number, s_unit
cont>            from tom
cont>           where (s_line = 0 or s_line is NULL)
cont>  do
```

```
cont>    trace :a.s_number,'-'||'>', :a.s_unit;
cont>  end for;
cont> end;
~Xt: 1         ..1
~Xt: 2         ..2
~Xt: 3         ..3
SQL>
```

This problem was caused by an interaction with the Rdb optimizer when it was processing a range list strategy for the outer FOR loop. If the OR condition is removed from the example query, the results of CONCAT are correct.

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.2.5 CREATE INDEX Inserts Index Key in Wrong Partition

Bug 2797443

In prior releases of Oracle Rdb, in rare cases, creating a partition index with a descending CHAR or VARCHAR column that had a collating sequence might map rows to the wrong partition. Such a definition may appear as follows:

```
create collating sequence GERMAN german;
create domain NAMES_DOM char(40) collating sequence GERMAN;
create domain IDS_DOM char collating sequence GERMAN;
create table NAMES_TABLE (id IDS_DOM, last_name NAMES_DOM);
create index NAMES_INDEX
        on NAMES_TABLE (id, last_name)
        store using (id, last_name)
            in AREA1 with limit of ('1', 'k')
            in AREA2 with limit of ('1', 'm')
            otherwise AREA3;
```

The problem does not exist if the rows are inserted after the index is created or if the collating sequence is removed.

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

1. The columns have an associated collating sequence. This may be explicitly defined on the domains or implicitly as a database wide collating sequence.
2. The data rows are inserted first before the index is created with two or more segments, some of which are descending.
3. The STORE USING statement contains WITH LIMIT OF clauses where the leading segment (or segments) are the same, but the values of the second segment are descending.

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.2.6 Unexpected Bugcheck When Inserting Into a View

Bug 4001764

In prior releases of Oracle Rdb V7.2, using a view that referenced an AUTOMATIC AS column or a column with a DEFAULT value could cause a bugcheck. This occurred when the referenced column also referenced

other columns from the base table. The following example shows the problem.

```
SQL> create table sample_auto
cont>     (cola integer(3)
cont>     ,colb automatic insert as round(cola));
SQL> insert into sample_auto values (18.245);
1 row inserted
SQL>
SQL> create view sample_view as select * from sample_auto;
SQL>
SQL> insert into sample_view values (19.456);
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000000000010, PC=0000000000288F7C, PS=0000001B
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.1.3. Oracle Rdb now correctly resolves the reference to the AUTOMATIC AS column and the DEFAULT value clause from within a view.

4.2.7 SQLSTATE 22001 Not Returned with Dynamic SQL

Bug 5208854

When using the dialects SQL92, SQL99, ORACLE LEVEL1, or ORACLE LEVEL2, SQL is required to report "22001" in the SQLSTATE field along with a negative SQLCODE when a value input into a SQL statement is truncated.

If a Precompiled SQL program used a variable to provide an input value for a SQL statement and this variable was longer than the target database field and contained non−blank data beyond the size of the field, the required SQLSTATE and SQLCODE were not being reported.

For example, consider a database table named CH1 with the following columns defined:

```
Column Name                    Data Type
-----------                    ---------
CH1A                           CHAR(10)
CH1B                           CHAR(1)
CH1C                           CHAR(10)
```

The following fragment of a Precompiled SQL/C program contains the following insert code:

```
...
    long SQLCODE;
    char SQLSTATE[6] = {'\0','\0','\0','\0','\0','\0'};
    char dstmt[100];
    char x4[16];
...
    strcpy(dstmt, "SET DIALECT 'SQL92'");
    EXEC SQL EXECUTE IMMEDIATE :dstmt;

    strcpy (dstmt, "INSERT INTO CH1 VALUES ('FOO', 'F', ?)          ");
    EXEC SQL PREPARE BLAT4 FROM :dstmt;

    strcpy (x4, "LITTLETOOLONG");
    EXEC SQL EXECUTE BLAT4 USING :x4;
```

```
    printf("SQLCODE should be < 0, SQLSTATE should be 22001\n")
    printf("SQLCODE is %ld\n", SQLCODE);
    SQLSTATE[5] = '\0';
    printf ("SQLSTATE is %s\n", SQLSTATE);
...
```

The output of this program would have appeared as:

```
SQLCODE should be < 0, SQLSTATE should be 22001
SQLCODE is 0
SQLSTATE is 00000
```

The output of the program should be:

```
SQLCODE should be < 0, SQLSTATE should be 22001
SQLCODE is −306
SQLSTATE is 22001
```

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.2.8 Unexpected Bugcheck When Creating a Duplicates Allowed SORTED RANKED Index

Bug 3899550

In prior versions of Oracle Rdb, the CREATE INDEX command may fail for SORTED RANKED indices if the number of duplicates is very high (larger than 16777217). The bugcheck dump footprint looks similar to the following:

- COSI−F−BUGCHECK, internal consistency failure
- Exception occurred at PSIIBUILD2CREATENODE + 0000066C
- Called from PSIIBUILD2BUILDFROMBOTTOM + 00000A2C
- Called from PSII2CREATETREE + 00000244
- Called from RDMS$$KOD_CREATE_TREE + 000001E4
- Bugcheck when working on sorted ranked index

This problem has been corrected in Oracle Rdb Release 7.2.1.3. Oracle Rdb now accumulates the leaf node cardinalities in larger capacity internal counters.

4.2.9 SQLSTATE 22011 and SQLCODE −1044 Added

When using the dialects SQL92, SQL99, ORACLE LEVEL1, or ORACLE LEVEL2, SQL is required to report a negative SQLCODE and a SQLSTATE of "22011" (meaning "Invalid substring") as a result of a parameter error in a SUBSTRING built−in function. SQL was returning the generic error result, that is, a SQLCODE of −1 and SQLSTATE of "RR000".

For example, the following query against a PERSONNEL database fails due to a negative string length.

```
SQL> select substring(last_name from 1 for −1) from employees;
%RDB−F−INVSUBSTRLEN, length specified for substring is invalid
```

Oracle Rdb has been enhanced so that for such failures it will produce a SQLCODE of −1044 (Meaning "Invalid substring length") and the SQL Standard−required SQLSTATE of "22011". For example, the SQLCA now appears as follows after the above query:

```
SQL> show sqlca
SQLCA:
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        −1044
        SQLERRD:        [0]: 3
                        [1]: 0
                        [2]: 0
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       22011
```

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.2.10 SUBSTRING Truncation for View Defined in Program

Bug 6117796

A precompiled SQL or SQL Module language program which defined a view containing a column defined by using a SUBSTRING built−in function would receive truncated results in some cases when using the view for retrievals during the same program execution.

For example, suppose a Precompiled SQL program contained the following SQL statements:

```
EXEC SQL CREATE TABLE MOREGRUB (C1 VARCHAR (10), ID INT);
EXEC SQL CREATE VIEW X4 (S1, ID) AS
  SELECT (C1 FROM 2 FOR 4), ID
  FROM MOREGRUB;
EXEC SQL INSERT INTO MOREGRUB VALUES ('Pretzels', 1);
EXEC SQL SELECT S1 INTO :ch1 FROM X4 WHERE ID = 1;
```

After the above code was executed, the value in the host variable ":ch1" would be "re " instead of the correct value of "retz".

The problem can be worked around by defining the view external to the program or by selecting directly against the table instead of using a view.

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.2.11 Unexpected PORT_LEN Error When Calling Storage Mapping Function

Bug 6129632

In prior releases, Oracle Rdb would create a special storage map function using the same name as the table's storage map. This function can be passed data values and have the partition number returned to the caller. In

some cases, such calls will fail with an RDB−F−PORT_LEN error as shown in this example.

```
SQL> SELECT sample_map (acct_num) FROM sample;
%RDB−F−PORT_LEN, buffer length 25 does not match BLR description 29
```

This error occurs when the total length of the user data is a multiple of 4. In this case, the column ACCT_NUM is of type INTEGER (that is 4 octets in length). There is no workaround for this problem.

Once this release of Oracle Rdb has been installed, the following command can be used to regenerate a working function definition:

```
SQL> ALTER STORAGE MAP sample_map COMPILE;
SQL> COMMIT;
```

This command should be applied to any storage map that exhibits this error.

This problem has been corrected in Oracle Rdb Release 7.2.1.3. Oracle Rdb now correctly generates the storage mapping function.

4.2.12 Unexpected Failure of ALTER DATABASE to Enable PERSONA Support

Bug 6132645

In some cases, an ALTER DATABASE ... SECURITY CHECKING IS EXTERNAL (PERSONA IS ENABLED) would fail to enable PERSONA on the database. This happens in rare cases when the database has a very low number of buffers. In the reported case, the database was defined with NUMBER OF BUFFERS 10.

The workaround is to increase the number of buffers for the database to 100 or so to allow the buffer containing the RDB$DATABASE row to remain in memory.

This problem has been corrected in Oracle Rdb Release 7.2.1.3. This release of Rdb ensures that the RDB$DATABASE row is current during the update of the RDB$FLAGS column.

4.3 RMU Errors Fixed

4.3.1 RMU Backup To Tape With /ENCRYPTION Can Cause Bugcheck

Bug 6196884

An RMU Backup to tape operation using a non−zero XOR group count and encryption can fail with an access violation error. See the following example.

```
$ RMU/BACKUP/ENCRYPTION=(VAL="mysecretkey") MF_PERSONNEL MKA0:TEST.RBF
...
%RMU−I−BUGCHKDMP, generating bugcheck dump file ...
%SYSTEM−F−ACCVIO, access violation, reason mask=04, virtual address=...
%RMU−F−FATALERR, fatal error on BACKUP
```

The problem was caused by the encryption code using an incorrect buffer size which had been modified by the XOR code.

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.3.2 RMU /ANALYZE /INDEX Accesses All Logical and Physical Areas

Bug 3146660

In previous Oracle Rdb releases, the RMU /ANALYZE /INDEX command would attempt to access all logical and physical areas of the database. This would result in excessive I/O and locking operations and could cause unexpected logical or physical area lock conflicts with other database users.

This problem has been corrected in Oracle Rdb Release 7.2.1.3. The RMU /ANALYZE /INDEX command now only accesses those logical and physical areas for the indexes being analyzed.

4.3.3 %COSI−F−NEGTIM Could Abort RMU/VERIFY Or RMU/BACKUP

Bug 6005440

There was a problem where the following message

```
%COSI−F−NEGTIM, a negative time was computed
```

could abort the RMU/BACKUP or RMU/VERIFY of an Oracle Rdb database. This happened during the Oracle Rdb database root verification executed by the RMU/VERIFY and RMU/BACKUP commands if the Oracle Rdb database root creation time was outside the allowable range. The %COSI−F−NEGTIM error was allowed to abort the RMU/BACKUP or RMU/VERIFY operation when the command execution should have been allowed to continue.

One possible cause of this problem besides database root corruption would be if the VMS system time was set to a time earlier than the database creation time.

This problem has been corrected so that if a %COSI−F−NEGTIM error does occur when the database root is verified, the RMU/VERIFY or RMU/BACKUP will be allowed to continue.

The following example of this problem happens during the verification of the database root at the beginning of the backup of an Oracle Rdb database. After the %RMU−W−ROOTADINV warning is output because of an invalid creation date value in the database root, the %COSI−F−NEGTIM error aborts the backup operation.

```
$rmu/backup mf_personnel mfp.rbf
%RMU−W−ROOTADINV, root "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1",
                  contains incorrect time stamp
                  expected between  1−DEC−1981 00:00:00.00 and
                  1−JAN−2007 00:00:01.24,
%COSI−F−NEGTIM, a negative time was computed
%RMU−F−FATALOSI, Fatal error from the Operating System Interface.
%RMU−F−FTL_BCK, Fatal error for BACKUP operation at  1−JAN−2007 00:00:02.59
```

If this problem is caused by the VMS system time being set to a time earlier than the database creation time, a workaround for this problem would be to set the system time to a time after the database creation. Otherwise, RMU/RESTORE/ONLY_ROOT could be used to restore a valid database root from a previous database backup.

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.3.4 RMU/VERIFY Problem With Database Creation Date Diagnostics

There was a problem that if either of the following diagnostic messages was issued during the database root verification executed by the RMU/VERIFY and RMU/BACKUP commands (due to the fact that the Rdb database root creation time was zero or outside the allowable range), incorrect diagnostic messages or incorrect date values in diagnostic messages could be output later on. This is because the root creation time was not correctly saved for later use to verify AIJ and page timestamps if one of these errors was output.

```
%RMU−W−ROOTADINV, root "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1",
                  contains incorrect time stamp
                  expected between  1−DEC−1981 00:00:00.00 and
                  1−JAN−2007 00:00:02.54,
                  found: 19−APR−2007 03:35:27.81

%RMU−W−ROOTADZER, root "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1",
                  contains zero time stamp
```

This problem has been corrected so that even though the root creation time is zero or outside of the allowable range, it will be saved correctly for use in later diagnostics.

One possible cause of this problem besides database root corruption would be if the VMS system time was set to a time earlier than the database creation time.

The following example of this problem happens during the verification of the database root at the beginning of the backup of an Oracle Rdb database. After the %RMU−W−ROOTADINV warning is output, the later %RMU−E−BADTADAIJ error diagnostic contains an uninitialized VMS system date of "17−NOV−1858

00:00:00.00".

```
$rmu/backup mf_personnel mfp.rbf
%RMU-W-ROOTADINV, root "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1",
                  contains incorrect time stamp
                  expected between  1-DEC-1981 00:00:00.00 and
                  1-JAN-2007 00:00:01.24,
                  found:  1-JAN-2007 00:01:29.14
%RMU-E-BADTADAIJ, after-image journal creation version differs from the root
                  expected: 17-NOV-1858 00:00:00.00, found: 1-JAN-2007
    00:01:29.14
%RMU-W-ROOERRORS,        1 error encountered in root verification
%RMU-I-COMPLETED, BACKUP operation completed at  1-JAN-2007 00:00:01.40
```

If this problem is caused by the VMS system time being set to a time earlier than the database creation time, a workaround for this problem would be to set the system time to a time after the database creation. Otherwise RMU/RESTORE/ONLY_ROOT could be used to restore a valid database root from a previous database backup.

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.3.5 RMU/RESTORE Exits with Traceback Log When the Wrong Version is Used

Bug 6001235

Using RMU/RESTORE to restore an incompatible version of a database exits with a register dump instead of simply exiting. This can be seen in the following example.

```
$ RMU/RESTORE/NOCDD/DIR=SYS$DISK:[]/LOG REPRODUCER.RBF
%RMU-F-DB_CVT_FAIL, Cannot convert from version V7.2 to V7.1
%RMU-F-DB_CVT_FAIL, Cannot convert from version V7.2 to V7.1

  Improperly handled condition, image exit forced by last chance handler.
    Signal arguments:   Number = 0000000000000008
                        Name   = 0000000002C88B94
                                 0000000000000004
                                 0000000000000007
                                 0000000000000002
                                 0000000000000007
                                 0000000000000001
                                 00000000003AEB34
                                 100000000000001B
  ...
```

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.3.6 RMU/VERIFY %RMU−I−BADNXTNOD Diagnostic Message Changed To %RMU−E−BADNXTNOD

Bug 4197298

The %RMU−I−BADNXTNOD diagnostic message that could be displayed when verifying Oracle Rdb

database sorted indexes indicated that the index structure was corrupt and that the index needed to be rebuilt. Therefore the BADNXTNOD diagnostic message severity level has been changed from Informational to Error so that this message will not be ignored.

The following example shows the %RMU−I−BADNXTNOD diagnostic message that was put out when RMU/VERIFY detected a corruption in the structure of a sorted index in an Rdb database. This indicated that the index needed to be rebuilt.

```
$ RMU/VERIFY/INDEX=bad_index mf_personnel
%RMU−I−BADNXTNOD, Bad next b−tree node at level 2.
                  Expected b−tree node at logical dbkey 39:3:2.
                  Found next b−tree node at logical dbkey 65:3:2.
```

The following example shows the %RMU−E−BADNXTNOD diagnostic message which will now be output with an "Error" instead of an "Informational" severity to make sure that the index corruption is brought to the user's attention.

```
$ RMU/VERIFY/INDEX=bad_index mf_personnel
%RMU−E−BADNXTNOD, Bad next b−tree node at level 2.
                  Expected b−tree node at logical dbkey 39:3:2.
                  Found next b−tree node at logical dbkey 65:3:2.
```

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.3.7 RMU/VERIFY %RMU−I−BTRDUPCAR Diagnostic Message Changed To %RMU−E−BTRDUPCAR

The %RMU−I−BTRDUPCAR diagnostic message that could be displayed when verifying Oracle Rdb database Sorted Ranked indexes indicates that the index structure is corrupt and that the index needs to be rebuilt since the cardinality specified in an index entry is inconsistent with the cardinality computed from the duplicate list for the entry. This will cause wrong results for SQL queries like COUNT(*). Therefore, the RMU−I−BTRDUPCAR diagnostic message severity level has been changed from Informational to Error so that this message will not be ignored.

The following example shows the %RMU−I−BTRDUPCAR diagnostic message that was put out when RMU/VERIFY detected a corruption in the structure of a Sorted Ranked index in an Rdb database. This indicated that the index needed to be rebuilt.

```
$ RMU/VERIFY/INDEX=bad_index mf_personnel
%RMU−I−BTRDUPCAR, Inconsistent duplicate cardinality (C1) of 16777318  specified
                  for entry 1 at dbkey 58:10:0.
                  Actual count of duplicates is 16777317.
%RMU−I−BTRROODBK, root dbkey of B−tree is 58:10:0
```

The following example shows the %RMU−E−BTRDUPCAR diagnostic message which will now be output with an "Error" instead of an "Informational" severity to make sure that the index corruption is brought to the user's attention.

```
$ RMU/VERIFY/INDEX=bad_index mf_personnel
%RMU−E−BTRDUPCAR, Inconsistent duplicate cardinality (C1) of 16777318  specified
                  for entry 1 at dbkey 58:10:0.
                  Actual count of duplicates is 16777317.
%RMU−I−BTRROODBK, root dbkey of B−tree is 58:10:0
```

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.3.8 Possible Invalid %RMU−F−NOPRIVERR Error on RMU/RESTORE, RMU/DUMP/BACKUP

Bug 6119717

Because of a problem initializing the required VMS privileges needed to execute the RMU full and /ROOT_ONLY RESTORE commands and the RMU/DUMP/BACKUP command, an invalid %RMU−F−NOPRIVERR could occur because the initialization problem could make RMU falsely expect that certain VMS privileges such as WORLD access were required when they were not. This could happen only if the user process was non−privileged and therefore was not granted VMS override privileges such as BYPASS and SYSPRV. This was not a problem with checking the access rights in the database Access Control List, which was handled correctly.

The following example shows that an unprivileged user process was denied the right to execute the RMU/RESTORE command even though the database Access Control List saved in the RBF file allowed the user access to the database and the user held the required VMS privileges to allow execution of the RMU/RESTORE command.

```
$ SHOW PROCESS/PRIVILEGE

Authorized privileges:
 GROUP        GRPNAM       GRPPRV       NETMBX       TMPMBX

$ RMU/SHOW PRIVILEGE MF_PERSONNEL.RDB

    (IDENTIFIER=[GROUP,USER],ACCESS=READ+WRITE+CONTROL+RMU$ALL)

$ RMU/BACKUP/NOLOG MF_PERSONNEL.RDB MFP.RBF
$ SQL
 drop database filename mf_personnel;
 exit
$ RMU/RESTORE/NOLOG MFP.RBF
 %RMU−F−NOPRIVERR, no privileges for attempted operation
```

A possible workaround for this problem is to execute the RESTORE using a VMS account that is authorized with BYPASS or SYSPRV privileges.

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.3.9 RMU/REPAIR to Move Snapshot File Can Fail With Bad New File Specification

Bug 4083632

A concealed device name in the default file specification for a snapshot file and specifying a new snapshot location can cause the RMU/REPAIR to fail. See the following example.

```
$ RMU/REPAIR/INITIALIZE=(SNAPSHOT=CONFIRM)/AREA=MYDB_DATA −
    $1$DKA100:[USER.][SUB]MYDB
Area MYDB_DATA snapshot filename [$1$DKA100:[USER.][SUB]MYDB_DATA.SNP;1]:
```

```
    >>> User response: $1$DKA100:[USER]MYDB_DATA.SNP
Area MYDB_DATA snapshot file allocation [100]:
%RMU-F-FILACCERR, error creating database file $1$DKA100:[USER.]
[USER]MYDB_DATA.SNP;1
-RMS-E-DNF, directory not found
```

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.3.10 RMU−F−FILACCERR, Error Truncating File on RMU/BACKUP/AFTER

Bug 5629652

An RMU/BACKUP/AFTER of a single extensible after image journal file that had extended and then had been disabled prior to the backup of the journal file failed with the following error:

```
%RMU-F-FILACCERR, error truncating file
 -SYSTEM-W-ACCONFLICT, file access conflict
```

The following example shows the RMU−F−FILACCERR which occurred during the RMU/BACKUP/AFTER of a single extensible after image journal file that had extended and had then been disabled prior to the backup of the journal file.

```
$ SQL
create data file foo;
create table tbl1 (c1 char(50),c2 char(50));
commit;
exit
$ rmu/backup foo foo
$ rmu/set after/enable/add=(name=aij1,file=aij1) foo
$ SQL
att 'f foo';
@load.sql
exit
$ rmu/set after/disable foo
$ rmu/backup/after/log foo fooaij
   %RMU-F-FILACCERR, error truncating file
   -SYSTEM-W-ACCONFLICT, file access conflict
```

A workaround for this problem is not to disable the journal file prior to the backup of the journal file.

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.4 RMU Show Statistics Errors Fixed

4.4.1 Incorrect RMU/SHOW STATISTICS Transaction Recovery Duration Estimates

Bug 6006427

In previous Oracle Rdb 7.2 releases, it was possible for the time values on the Transaction Recovery Duration Estimates display to be wildly inaccurate. The values were generally vastly larger than they should have been.

This problem has been corrected in Oracle Rdb Release 7.2.1.3. The estimates are now scaled to more realistic range values.

4.4.2 RMU/SHOW STATISTICS Stall Statistics Aggregate Duration Incorrectly Scaled Values

Bug 6006202

In previous Oracle Rdb 7.2 releases, stall duration values on the aggregate stall statistics display were not correctly scaled and therefore were displayed as exceptionally large values.

This problem has been corrected in Oracle Rdb Release 7.2.1.3.

4.4.3 RMU /SHOW STATISTICS /ALARM=n Not Waiting n Seconds

Bug 6016126

In previous Oracle Rdb 7.2 releases, it was possible for stall notification alarms to fire inaccurately. Typically, the stall notification was much faster than expected. For example, the following should alarm via OPCOM if a stall longer than 60 seconds is detected. Alarms could be generated much earlier than they should have been.

```
$ RMU /SHOW STATISTICS PRODUCTION_DB -
    /NOINTERACTIVE -
    /BROADCAST -
    /TIME=60 -
    /ALARM=60 -
    /NOTIFY=(OPER12) -
    /UNTIL=TOMORROW
```

This problem has been corrected in Oracle Rdb Release 7.2.1.3. Stall notification alarms are not generated early unexpectedly.

4.4.4 State Value Truncated on Hot Standby Statistics Display

Bug 6044632

Previously, it was possible when using a Hot Standby TCP/IP port number greater than 9999 that the port number would be truncated on the RMU /SHOW STATISTICS Hot Standby Statistics display.

For example, when using a TCP/IP port number of 12345, the state display could be shown as "State: TCP/IP:1234"

This problem has been corrected in Oracle Rdb Release 7.2.1.3. The state display field now allows a 5−digit TCP/IP port number to be displayed.

4.5 Hot Standby Errors Fixed

4.5.1 Hot Standby Node Failure Recovery When Using RMU/OPEN/ROW_CACHE=DISABLE

Bug 5957364

In configurations using the Row Cache and Hot Standby features, row caching must be explicitly disabled on the standby database using the RMU/SET ROW_CACHE/DISABLE command prior to starting Hot Standby for the first time on the database. However, it is also possible (though not recommended) to use the RMU/OPEN/ROW_CACHE=DISABLE command on the standby database in order to suppress row caching.

When using the RMU/OPEN/ROW_CACHE=DISABLE command, if a system failure occurred, it was possible that the database recovery upon reopening the database would attempt to start with a very old last checkpoint location. This location was based on the row cache checkpoint from when the master database had been originally backed up to create the standby database. In some cases, the required AIJ files would no longer be online and the recovery would fail.

This problem has been corrected in Oracle Rdb Release 7.2.1.3. The DBR process now ignores the row cache oldest checkpoint location when not recovering from a node failure when the RCS process had been active.

Chapter 5
Software Errors Fixed in Oracle Rdb Release 7.2.1.2

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.1.2.

5.1 Software Errors Fixed That Apply to All Interfaces

5.1.1 Bugchecks at KOD$START + 0000080C

Bug 5059527

Beginning in Oracle Rdb Release 7.1.4.3, it was possible to get bugchecks with the following exception:

```
***** Exception at 0193578C : KOD$START + 0000080C
%COSI-F-BUGCHECK, internal consistency failure
```

In Release 7.1.4.3, the routine KOD$START was modified to verify that it was not assigned a TSN of zero. If a TSN of zero was assigned then this bugcheck would occur.

This problem was caused by a small race condition in the code responsible for determining the oldest transaction sequence number (TSN) in the database. Occasionally, if multiple processes were accessing the global oldest TSN location at the same time, the code would incorrectly determine that the oldest TSN was zero.

The only way to completely avoid the problem is to use a previous version of Oracle Rdb. The incidence can be reduced by setting the number of cluster nodes for the database to a value greater than one. Note that by doing so performance features that rely on the number of nodes being one, such as row cache, are disabled. Also, if the system is not part of a cluster, then setting the number of cluster nodes to a value greater than one will have no effect.

This problem has been corrected in Oracle Rdb Release 7.2.1.2. The race condition that led to a TSN of zero being assigned has been eliminated.

5.1.2 Adding a Large AIJ File to a DB Fails With Either an ACCVIO or OPCDEC Error

Bug 5852864

Creating an after image journal file for a database fails with either an ACCVIO or an OPCDEC error. See the following example.

```
SQL> CREATE DATABASE
     ...
     PRESTARTED TRANSACTIONS ARE on
        (WAIT 1 MINUTES FOR TIMEOUT)
     ...
SQL> ALTER DATABASE FILENAME myDB ADD JOURNAL journal01 FILENAME 'JOURNAL01'
ALLOCATION IS 7000000 BLOCKS;
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
                  virtual address=000000000000000C, PC=000000000000000C,...
```

This was caused by a bug in the code which prematurely cleared a synchronization flag. This allowed the request created by the expired prestarted transactions timer to execute before the current request had completed. The side effect of this was a stack corruption.

As a workaround, use a larger prestarted transactions timer value or disable the prestarted transactions timer completely using the SQL ALTER DATABASE statement.

This problem has been corrected in Oracle Rdb Release 7.2.1.2.

5.1.3 Simple Query Fails with ARITH_EXCEPT

Bug 5941200

A simple query fails with an arithmetic exception, as in the example below.

```
SELECT * FROM auth_control WHERE scheme_code = 'aaa'
AND task_ind = 'x'  AND in_use_ind = 'g';
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-SYSTEM-F-HPARITH, high performance arithmetic trap, Imask=00000000,
Fmask=00000400, summary=04, PC=00000000008BADFC, PS=0000001B
-SYSTEM-F-FLTDIV, arithmetic trap, floating/decimal divide by zero at
PC=00000000008BADFC, PS=0000001B
```

The query works if the SQL flag 'index_column_group' is disabled, as in the following example.

```
set flags 'noindex_column_group';
set flags 'stra,detail';

SELECT * FROM auth_control WHERE scheme_code = 'aaa'
AND task_ind = 'x'  AND in_use_ind = 'g';
0 rows selected
```

The query also works if RMU/COLLECT OPTIMIZER is applied, as in the following example.

```
$rmu/collect optimizer <database>
$SQL$
ATT 'FILE <database>';

SELECT * FROM auth_control WHERE scheme_code = 'aaa'
AND task_ind = 'x'  AND in_use_ind = 'g';
0 rows selected
```

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

1. The query selects from a table with three equality predicates, for example, COL1 = 1 AND COL2 = 2 AND COL3 = 3.
2. The table contains an index with all the above three columns in a contiguous order, for example, COL1, COL2, COL3, COL4.
3. One of the segment columns contains a zero value for the prefix cardinality. To find out this information, you need to set the SQL flags 'costing' and 'cursor_stat', and execute the query again. Here is the information on the prefix cardinality dumped out by the query:

   ```
   Segment group factors of 4 segments are estimated.
      COL1     GROUP_FACTOR  3.2407680E+00     PREFIX_CARD 1
   ```

```
COL2      GROUP_FACTOR  2.4153826E+00     PREFIX_CARD 1
COL3      GROUP_FACTOR  1.8002133E+00     PREFIX_CARD 0
COL4      GROUP_FACTOR  1.3417203E+00     PREFIX_CARD 0
```

This problem affects the following Oracle Rdb releases: V7.1.4.4, V7.1.4.5, V7.2.0.1, V7.2.0.2 and V7.2.1.0. This problem has been corrected in Oracle Rdb Release 7.2.1.2.

5.2 SQL Errors Fixed

5.2.1 Unexpected RTN_FAIL/BAD_REQ_HANDLE Reported when Stored Function Called with All Constant Parameters

Bug 4764496

In prior releases of Oracle Rdb, it was sometimes possible for a query to fail with the error RTN_FAIL for "(unknown)" as the routine name, and a BAD_REQ_HANDLE secondary message.

The following shows an example using the REPLACE function from the SQL_FUNCTIONS routines.

```
select
  replace('hallo','a','e')
  from TAB1
 where COL2 = date'2006-06-29'
   and (  COL1 = 'AA'
          or
          COL1 = 'BB' );
%RDB-E-RTN_FAIL, routine "(unknown)" failed to compile or execute successfully
-RDB-E-BAD_REQ_HANDLE, invalid request handle
```

This problem occurred when the query optimizer tried to make use of the constant expression during query construction. Unfortunately, the referenced routine (in the example the function was named REPLACE) was not yet compiled for use by the query, and hence the name reported was unknown. Once the function is loaded this query will succeed.

Some workarounds for this problem include:

- Define the function with the NOT DETERMINISTIC option to prevent it being considered a constant expression in this case.
- Execute ALTER FUNCTION ... COMPILE to pre−load the function prior to executing queries that reference it.
- Execute ALTER MODULE ... COMPILE to pre−load all functions in a module.
- Replace the constant expression with a varying column value.

This problem has been corrected in Oracle Rdb Release 7.2.1.2.

5.2.2 SYSTEM−F−ROPRAND With Large Command Line

Bug 4117738

With a very large command line, it was possible to receive a SYSTEM−F−ROPRAND message with interactive SQL. Furthermore, SQL was left in a state such that all subsequent commands resulted in a SYSTEM−F−ROPRAND error. In the example in the bug report, a command file consisting of a single line of over 11,000 bytes caused the problem to manifest itself.

The following sequence shows the problem symptom using a command file named "BIGCMD.SQL" which is too large to reproduce here.

```
SQL> @BIGCMD.SQL
%SYSTEM-F-ROPRAND, reserved operand fault at PC=0000000000F4DDA8, PS=0000001B
```

Note that a variety of other symptoms could also occur with such a large command line such as an access violation.

Note also that even after the problem in SQL is corrected, the command line in the example above contained a query that needed levels of recursion greater than the default size of Rdb's executive mode stack. If this occurs, the query will fail with the following messages:

```
%RDB-F-IMP_EXC, facility-specific limit exceeded
-RDMS-F-XPR_STACK_OFLO, expression forces too many levels of recursion
```

The size of the executive mode stack can be adjusted using the logical RDMS$BIND_EXEC_STACK_SIZE. The default size is 20 pagelets and the query in the command file referred to above required the size to be set to 330 in order to complete.

This problem has been corrected in Oracle Rdb Release 7.2.1.2.

5.2.3 Unexpected Bugcheck from ALTER VIEW

Bug 5964501

In prior releases of Oracle Rdb, an ALTER VIEW command which specified an AS SELECT clause could bugcheck. This would occur when the column expression included a Boolean expression such as appears in a CASE, COALESCE, NULLIF, NVL, NVL2, ABS, DECODE, or subselect statement.

The following example shows the error.

```
SQL> alter view v_t1 as select nullif(col1,-999999) as nn from t1;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file
RDBVMS_USER2:[SMITHI]RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file
RDBVMS_USER2:[SMITHI]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=
000000000000000C, PC=000000000028878C, PS=0000001B
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.1.2. In addition, Oracle Rdb now correctly stores the table dependency information in RDB$VIEW_RELATIONS system table for the view.

5.2.4 Unexpected Bugcheck from ALTER INDEX ... BUILD PARTITION

Bug 5951999

In prior releases of Oracle Rdb, it was possible that an ALTER INDEX ... BUILD PARTITION or ALTER INDEX ... REBUILD PARTITION statement on a SORTED RANKED index would bugcheck. This problem does not affect HASHED or SORTED indices.

The bugcheck is initiated by a sanity check within the SORTED RANKED index facility when it detects that

database keys for the partition were not sorted within duplicate index keys.

The workaround for this problem is to use the REBUILD ALL PARTITIONS clause or to drop and re−create the index.

This problem has been corrected in Oracle Rdb Release 7.2.1.2. Rdb now correctly sorts the duplicate index key values so that they are ascending by DBKEY.

5.3 RMU Errors Fixed

5.3.1 RMU/DUMP/BACKUP Fails With an ACCVIO or an Overrun Error

Bug 5968242

The command RMU/DUMP/BACKUP for a save set saved with /COMPRESSION=ZLIB fails with the following error:

```
%SYSTEM-F-ACCVIO, access violation, reason mask=04, virtual address=...,
PC=..., PS=0000001B
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
```

The command RMU/DUMP/BACKUP for a save set saved with /COMPRESSION=HUFF fails with the following error:

```
Illegal output buffer overrun
%RMU-E-INVRECEXP, Error expanding compressed backup file record.
%RMU-F-FATALERR, fatal error on DUMP_BACKUP
```

The error occurred in a /DUMP specific code section which did not switch to the correct compression context for an area.

These problems have been corrected in Oracle Rdb Release 7.2.1.2.

5.3.2 RMU/RESTORE Convert Problem With More Than 32 Client Sequences

Bug 5963088

The Client Sequences count in the Rdb database root was arbitrarily set to 32 in the converted V7.2 database even if it was greater than 32 in the V7.1 database being converted to V7.2. This caused database corruption. This only happened when a V7.2 RMU/RESTORE of a V7.1 database RBF backup file caused the database to be converted to V7.2 at the end of the restore. It did not happen if the RMU/CONVERT command was used to convert a V7.1 database to V7.2. This problem will not happen if the count of Client Sequences does not exceed 32 in the V7.1 database being converted to V7.2. Unfortunately, the only way to eliminate this corruption is to repeat the conversion using one of the two workarounds described below. Note that this kind of problem shows the importance of backing up your V7.1 database before attempting to convert it to V7.2 and running RMU/VERIFY/ALL immediately following any database conversion.

The following example shows that the convert would complete without error but an RMU/VERIFY would show that the RDB$SEQUENCES system table contained more client sequences than were entered in the database root and an attempt to display the Client Sequences in SQL would cause an RDMS bugcheck dump.

```
$ RMU/RESTORE/LOG/DIR=DEVICE:[DIRECTORY]/nocdd -
_$ MFP_SEQUENCES.RBF
%RMU-I-CVTCOMSUC, CONVERT committed for DEVICE:[DIRECTORY]
 MF_PERSONNEL.RDB;1 to version V7.2
```

```
%RMU-W-USERECCOM, Use the RMU Recover command. The journals are not
available.
%RMU-I-COMPLETED, RESTORE operation completed at 30-MAR-2007 15:42:40.23
GIBSON>rmu/verify/all mf_personnel
%RMU-E-NOSEQENT, sequence id 35 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 37 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 38 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 39 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 40 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 41 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 42 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 43 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 44 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 45 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 46 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 47 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 48 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 49 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 51 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 52 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 53 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 54 has no valid entry in the root file
%RMU-E-NOSEQENT, sequence id 55 has no valid entry in the root file
$ SQL
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SHOW SEQUENCES;
Sequences in database with filename mf_personnel
     EMPID
     N1
     N10
     N11
     N12
     N13
     N14
     N15
     N16
     N17
     N18
     N19
     N2
     N20
     N21
     N22
     N23
     N24
     N25
     N26
     N27
     N28
     N29
     N3
     N30
%RDMS-I-BUGCHKDMP, generating bugcheck dump file
 DEVICE:[DIRECTORY]RDSBUGCHK.DMP;
```

One workaround for this problem is to do an RMU/CONVERT of the V7.1 database to V7.2 (see the example below). Another workaround is to drop the client sequences in the V7.1 database before backing it up, do the restore to V7.2, and then redefine the client sequences in the V7.2 database.

```
$ RMU/CONVERT MF_PERSONNEL
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.2-00
```

```
 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
%RMU-I-CVTCOMSUC, CONVERT
 committed for DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 to
 version V7.2
```

This problem has been corrected in Oracle Rdb Release 7.2.1.2.

5.3.3 RMU LOAD Delimited Text Problem Parsing NULL Values

Bug 5768090

There was a problem parsing values flagged as NULL in RMU UNLOAD and LOAD using DELIMITED TEXT delimiters that occurred when the character string used to designate a field flagged as NULL was the same as the combined character strings used to designate the PREFIX and SUFFIX delimeters which mark the start and end of column values. This problem was due to RMU/LOAD interpreting the NULL value as a PREFIX delimeter followed by a SUFFIX delimiter. This caused the column not to be flagged as NULL when it was loaded. This could cause errors on the RMU/LOAD. One example is an attempt to load an invalid date value if the field being loaded was a date field. It could also cause unintended values to be loaded to a field that was supposed to be flagged as NULL.

This problem has been fixed but we strongly recommend that a unique string or single character value for NULL be used that cannot be confused by the RMU UNLOAD and LOAD parsing with the prefix, the suffix or the prefix plus suffix single character or string values.

In ambiguous cases, where identical delimeters can be confused, the parser has to give precedence to one choice over another. It tries to pick the most common and reasonable case, but that may not always coincide with the original intent of the user. Using a unique NULL value or taking the default NULL value would avoid these cases.

The following shows one example of the case we have fixed where a problem occurred during the RMU/LOAD because the NULL string equaled the combined values of the prefix and the suffix character strings, which in this case each consisted of a single character. This case might not be obvious to a user because the prefix and suffix characters were not specified but the documented default prefix and suffix single character delimeter values of " were used. Since the NULL string was specified as "", RMU/LOAD interpreted the NULL value "" as a default prefix value " followed by a default suffix value ". This caused the NULL bit not to get set for the middle columns. Note that as documented

```
NULL="""""""
```

designates a NULL designator value of "" since the two outer double quote characters are stripped and each " character must be entered as "".

```
$ SQL
create database filename test;
create table t1 (col1 char(5), col2 date vms, col3 char(5));
create table t2 (col1 char(5), col2 integer, col3 char(5));
```

```
create table t3 (col1 char(5), col2 char(5), col3 char(5));
commit;
insert into t1 (col1,col3) values ('first','last');
insert into t2 (col1,col3) values ('first','last');
insert into t3 (col1,col3) values ('first','last');
commit;
SQL> sel * from t1;
 COL1    COL2                      COL3
 first   NULL                      last
1 row selected
SQL> sel * from t2;
 COL1           COL2   COL3
 first          NULL   last
1 row selected
SQL> sel * from t3;
 COL1    COL2    COL3
 first   NULL    last
1 row selected
 Exit
$ rmu/unload/record=(file=t1,format=delimited,null="""""""") test t1 t1
$ rmu/unload/record=(file=t2,format=delimited,null="""""""") test t2 t2
$ rmu/unload/record=(file=t3,format=delimited,null="""""""") test t3 t3
$ type t1.unl
"first","","last "
$ type t2.unl
"first","","last "
$ type t3.unl
"first","","last "
```

Because RMU confused the NULL value for the PREFIX plus SUFFIX values, the RMU/LOAD fails for the date field, and inserts a zero in the case of an integer (or real), and an empty string in the case of a character field.

```
$ rmu/load/record=(file=t1,format=delimited,null="""""""") test t1 t1
%RMU-I-LOADERR, Error loading row 1.
%RDB-E-CONVERT_ERROR, invalid or unsupported data conversion
-COSI-F-IVTIME, invalid date or time
%RMU-I-DATRECREAD,  1 data records read from input file.
%RMU-I-DATRECSTO,   0 data records stored.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 18-JAN-2007 04:23:54.48
$ rmu/load/record=(file=t2,format=delimited,null="""""""") test t2 t2
%RMU-I-DATRECREAD,  1 data records read from input file.
%RMU-I-DATRECSTO,   1 data records stored.
$ rmu/load/record=(file=t3,format=delimited,null="""""""") test t3 t3
%RMU-I-DATRECREAD,  1 data records read from input file.
%RMU-I-DATRECSTO,   1 data records stored.
$ SQL
SQL> att 'f test';
SQL> sel * from t1;
 COL1    COL2                      COL3
 first   NULL                      last
1 row selected
SQL> sel * from t2;
 COL1           COL2   COL3
 first          NULL   last
 first             0   last
2 rows selected
SQL> sel * from t3;
 COL1    COL2    COL3
 first   NULL    last
 first           last
```

```
2 rows selected
```

A workaround for this problem is to designate a unique value for NULL that cannot be confused with the default or specified values of other DELIMITED TEXT delimeters.

This problem has been corrected in Oracle Rdb Release 7.2.1.2.

5.4 RMU Show Statistics Errors Fixed

5.4.1 RMU/SHOW STATISTICS AIJ ARB:I/O Ratio, Blocks−per−I/O Ratio Problems

There was a problem with detecting the warning thresholds set for the "Examine ARB:I/O ratio" and "Examine blocks−per−I/O ratio" options on the "AIJ Analysis" screen display produced by the RMU/SHOW STATISTICS command for Oracle Rdb databases with after image journaling enabled. This problem caused the "ARB:I/O ratio" warning

```
#.# ARBs per I/O below #.# threshold
```

to not always be output when the currently set threshold for this ratio was passed and caused the "blocks−per−I/O ratio" warning

```
#.# blocks written per I/O below #.# threshold
```

to not always be output when the currently set threshold for this ratio was passed. This problem was caused by treating these thresholds as percent values instead of count values. This problem has been fixed and the misleading percent signs have been removed from these warning messages.

This problem has been corrected in Oracle Rdb Release 7.2.1.2.

5.5 Hot Standby Errors Fixed

5.5.1 Unable to Fully Disable Hot Standby Governor

Bug 5166721

Previously, when the Hot Standby Governor was explicitly disabled, it was still possible during periods of very high load for the Governor to be re−enabled when more than 75% of the LRS buffers on the standby system were in use.

This behavior has been disabled by default. The Governor will not be re−enabled even when more than 75% of the LRS buffers on the standby system are in use.

If it is desired to revert back to the prior behavior, the system−wide logical name RDM$BIND_LRS_ALLOW_AUTOMATIC_HOT_STANDBY_GOVERNOR may be defined to a value of "1" to allow the Governor to re−enable itself as in prior releases.

This problem has been corrected in Oracle Rdb Release 7.2.1.2.

Chapter 6
Software Errors Fixed in Oracle Rdb Release 7.2.1.1

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.1.1.

6.1 Software Errors Fixed That Apply to All Interfaces

6.1.1 Unexpected Query Failure With RDB−E−NO_RECORD Error

Bug 5846989

In prior releases of Oracle Rdb, it was possible for queries that used the MIN or MAX function to fail with the following error.

```
%RDB−E−NO_RECORD, access by dbkey failed because dbkey is
no longer associated with a record
−RDMS−F−NODBK, −524:22806:1 does not point to a data record
```

The DBKEY being displayed is an encoded DBKEY referencing the duplicate chain for the matching key.

This problem occurs under the following conditions:

- The query is solved using the Dynamic optimizer. This often means that Rdb has a choice of indices which could be used to solve the query.
- The selected index allows duplicate values.
- The minimum or maximum value contained more than one matching row.

As a workaround, the dynamic optimizer can be disabled using the logical name RDMS$SET_FLAGS or the SET FLAGS command with the "MAX_STABILITY" keyword, or by defining the logical name RDMS$MAX_STABILITY to the value 1.

An alternate workaround is to start the transaction using the clause ISOLATION LEVEL REPEATABLE READ.

This problem has been corrected in Oracle Rdb Release 7.2.1.1.

6.1.2 AIJ Backup Operation Aborts With NONAME−F−NOMSG Message Number 00000004

Bug 5890612

In rare cases, an after−image journal backup operation may fail with an unexpected incorrect status value. The actual value may vary, but at least one customer report of the problem indicated a value of 00000004. A bugcheck dump file "footprint" of this problem is:

```
***** Exception at 0054D94C : AIJBCK$GET_NEXT_JOURNAL + 00000CFC
Saved PC = 005452E8 : AIJBCK$FULL_BACKUP + 00000FF8
Saved PC = 00543C0C : AIJBCK$BACKUP + 0000113C
Saved PC = 0040E7A4 : RMUCLI$BACKUP_AIJ + 00000904
Saved PC = 003A2414 : RMU_DISPATCH + 00000484
Saved PC = 003A1AE8 : RMU_STARTUP + 000004E8
```

```
Saved PC = 001E002C : RMU$MAIN + 0000002C
```

This problem has been corrected in Oracle Rdb Release 7.2.1.1. The errant status value was the result of an uninitialized return status being passed back. The correct status is now returned.

6.1.3 Some RDMS$SET_FLAGS Settings Not Seen After ATTACH

Bugs 4103971, 4116768 and 5245116

Some database environment settings could be altered using a logical name or using the RDMS$SET_FLAGS logical name. In prior versions, the values set by the RDMS$SET_FLAGS logical were overwritten with default values.

The following example shows that some values are ignored.

```
$ define/nolog rdms$set_flags "MAX_STABILITY,NOINDEX_COLUMN_GROUP,
NOTRANSITIVITY,MAX_RECURSION(1000)"
$ SQL$
attach 'filename sql$database';
show flags;
Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(1000),REFINE_ESTIMATES(127),NOBITMAPPED_SCAN
$
```

This problem has been corrected in Oracle Rdb Release 7.2.1.1. The default value for the action is assigned prior to the processing of the RDMS$SET_FLAGS logical name.

6.1.4 RDB−E−EXCESS_TRANS Error After SET TRANSACTION Failure

Bug 5755008

Under some circumstances, a failure in a SET TRANSACTION statement with multiple databases, including at least one remote database, would leave that remote database in an indeterminate state. In this state, all SQL statements received an %RDB−E−EXCESS_TRANS error which named the remote database as having a transaction already in progress. The following ingredients were necessary to encounter the problem:

- Multiple databases must be attached, including at least one remote database.
- A transaction must be started for which Rdb does not use 2−phase commit. (For example, if all databases are read−only or if the logical SQL$DISABLE_CONTEXT is defined as TRUE, Rdb does not start a distributed transaction.)
- Rdb succeeds in starting a transaction on the remote database and then fails in starting the transaction on some other attached database.

In the above circumstance, Rdb will rollback the transaction for each of the databases for which it successfully started a transaction before failing on one of the databases. The most common reason for such a failure is a lock conflict with another process.

The following example illustrates the failure:

```
SQL> ATTACH 'ALIAS A1 FILENAME XENA::PERSONNEL';
SQL> ATTACH 'ALIAS A2 FILENAME PERSONNEL';
SQL> SET TRANSACTION ON A1 USING
cont>    (READ ONLY RESERVING A1.employees FOR SHARED READ)
cont>            AND ON A2 USING
cont>    (READ ONLY RESERVING A2.employees FOR SHARED READ,
cont>                         A2.bogus FOR SHARED READ);
%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-RELNOEXI, relation BOGUS
SQL>
SQL> show table a2.employees
%RDB-E-EXCESS_TRANS, exceeded limit of 1 transaction per database attachment
-RDB-F-ON_DB, on database DISK:[DIR]MF_PERSONNEL.RDB;1
```

In the above example, alias A1 is a remote database and alias A2 is a local database. Because the SET TRANSACTION statement specifies READ ONLY for both databases, Rdb does not use a distributed transaction (this is an optimization). The SET TRANSACTION statement fails because it tries to reserve a table named "bogus" in alias A2 which does not exist. This failure is normal and expected. But after the failure of the SET TRANSACTION, Rdb didn't properly roll back the transaction on alias A1. This caused the failure of the subsequent SHOW statement and all subsequent statements with an RDB-E-EXCESS_TRANS error.

The problem can be worked around by naming the remote alias last in the SET TRANSACTION statement.

This problem has been corrected in Oracle Rdb Release 7.2.1.1.

6.1.5 Float Overflow in Rdb Optimizer Cost Estimation

Bug 5727379

The following query generates the following float overflow error.

```
select * from game t1, post t2
  where (t1.post_ref_1 = t2.post_ref
      OR t1.post_ref_2 = t2.post_ref
      OR t1.post_ref_3 = t2.post_ref
      OR t1.post_ref_4 = t2.post_ref
      OR t1.post_ref_5 = t2.post_ref)
     AND t2.post_flag = 1
  limit to 1 row;
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-SYSTEM-F-HPARITH, high performance arithmetic trap, Imask=00000000,
Fmask=00000001, summary=02, PC=00000000007FB6F4, PS=0000001B
-SYSTEM-F-FLTINV, floating invalid operation, PC=00000000007FB6F4, PS=0000001B
```

The query works if the following predicate is changed using CAST.

```
select * from game t1, post t2
  where (t1.post_ref_1 = t2.post_ref
      OR t1.post_ref_2 = t2.post_ref
      OR t1.post_ref_3 = t2.post_ref
      OR t1.post_ref_4 = t2.post_ref
      OR t1.post_ref_5 = t2.post_ref)
     and CAST(t2.post_flag AS INTEGER) = 1   ! <==
```

```
  limit to 1 row;
0 rows selected
```

There is no known 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 joins two tables using the WHERE clause with multiple OR predicates and one AND filter predicate.
2. All the OR predicates contain the same column from the second table at the right hand side.

This problem has been corrected in Oracle Rdb Release 7.2.1.1.

6.1.6 Wrong Result From Query With Zig−zag Match and Reverse Scan

Bugs 5842319, 5850013, and 4771936

The following query with zig−zag match strategy using reverse scan returns the wrong result (should return one row).

```
SQL> set flags 'strategy,detail';
SQL> select d.trx_id
cont>  from t1 d
cont>  where exists (select * from t2 where trx_id = d.trx_id);
Tables:
  0 = T1
  1 = T2
Conjunct: <agg0> <> 0
Match
  Outer loop      (zig−zag)
    Index only retrieval of relation 0:T1
      Index name  T1_NDX [0:0]
  Inner loop      (zig−zag)
    Aggregate−F1: 0:COUNT−ANY (<subselect>)
    Index only retrieval of relation 1:T2
      Index name  T2_NDX [0:0]      Reverse Scan
0 rows selected
```

The tables contain the following data.

```
SQL> select * from t1;
 TRX_ID    SEQUENCE_NO
 0000044             1
 0000046             2
 0000047             1

SQL> select * from t2;
 TRX_ID    LOCATION_ID    COMPANY_NO
 0000046             5             2
 0000045             5             2
```

A workaround is to use SET FLAGS or define RDMS$SET_FLAGS to the value NOREVERSE_SCAN to disable the reverse scan feature. See the following example.

```
SQL> set flags 'noreverse_scan'
SQL> !... execute the same above query here ...
Tables:
  0 = T1
  1 = T2
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation 0:T1
      Index name  T1_NDX [0:0]
  Cross block entry 2
    Conjunct: <agg0> <> 0
    Aggregate-F1: 0:COUNT-ANY (<subselect>)
    Index only retrieval of relation 1:T2
      Index name  T2_NDX [1:1]
        Keys: 1.TRX_ID = 0.TRX_ID
 TRX_ID    SEQUENCE_NO
 0000046             2
1 row selected
```

This is problem is related to an incomplete fix for Bug 4771936. The key parts of this query which contributed to the error are:

1. The main select query joins two tables (in this example using an EXISTS clause) and a zig−zag match strategy is chosen for the solution.
2. The join key for the inner leg of the join is based on an index with DESC (descending) index segment but an ascending segment in the outer index.
3. A reverse scan is applied on the index retrieval at the inner leg.

This problem may occur under similar conditions for NOT EXISTS and normal join queries.

This problem affects the following Oracle Rdb Releases: V7.1.4.4, V7.1.4.5, V7.2.0.1, V7.2.0.2 and V7.2.1. This problem has been corrected in Oracle Rdb Release 7.2.1.1.

6.1.7 %RDB−E−REQ_NO_TRANS When Fetching From a Cursor

Bug 3000645

Under some circumstances, SQL would generate a %RDB−E−REQ_NO_TRANS error after the execution of a SQL stored procedure or multi−statement procedure which left the process with no currently active transaction. This error was often encountered in conjunction with using a hold cursor.

The following SQL creates a stored procedure (TX_END) which, if executed, would set up the state where the problem would have been encountered. Note that the stored procedure simply ends any active transaction and returns.

```
create module tx_end
    language sql
    declare transaction read only

procedure tx_end();
begin
declare :v_active tinyint  default 0;
get diagnostics :v_active = TRANSACTION_ACTIVE;
if (:v_active <> 0) then
```

```
    rollback;
end if;
end;
end module;
```

The problem could be worked around by ending the transaction with an explicit SQL statement in lieu of doing it inside a stored procedure.

This problem has been corrected in Oracle Rdb Release 7.2.1.1.

6.1.8 Distinct Query Bugchecks With Bitmap Scan Enabled

Bug 5295755

During the second execution of a "Distinct" query, with Bitmap Scan enabled, a bugcheck would occur.

```
set flags 'nobitmap,strategy,detail';
select  distinct test_nbr from t1 where asn='10000';
Tables:
  0 = T1
Reduce: 0.TEST_NBR
Leaf#01 Sorted 0:T1 Card=141
  Bool: 0.ASN = '10000'
  FgrNdx  IND_TN [0:0] Fan=17
  BgrNdx1 IND_ASN [1:1] Fan=14
    Keys: 0.ASN = '10000'
     TEST_NBR
        1.000
        2.000
2 rows selected

set flags 'bitmap,strategy,detail';
select  distinct test_nbr from t1 where asn='10000';
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
```

There were two exceptions possible:

```
***** Exception at 01001154 : RDMS$$EXE_CLOSE + 00000574
%COSI-F-BUGCHECK, internal consistency failure
```

or,

```
***** Exception at 0124631C : KOD$ROLLBACK + 0000032C
%COSI-F-BUGCHECK, internal consistency failure
```

The query works if the 'bitmap' is NOT enabled at the second execution, as in the following example.

```
set flags 'nobitmap,strategy';
select  distinct test_nbr from t1 where asn='10000';
Tables:
  0 = T1
Reduce: 0.TEST_NBR
Leaf#01 Sorted 0:T1 Card=141
  Bool: 0.ASN = '10000'
  FgrNdx  IND_TN [0:0] Fan=17
  BgrNdx1 IND_ASN [1:1] Fan=14
    Keys: 0.ASN = '10000'
```

```
     TEST_NBR
        1.000
        2.000
2 rows selected
```

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

1. The main select is a distinct query from a table with a filter predicate.
2. The table has two indices, each index contains one segment column.
3. The index that contains the column referenced by the filter predicate is applied as the background index.
4. The index that contains the column referenced by the distinct function is applied as the foreground index.
5. The SQL flag 'Bitmap' does not need to be enabled at the first execution, but it must be explicitly turned on before the second execution.
6. The strategy dynamic tactic must be "Sorted".
7. The background index scan completes sucessfully.

This problem affects the following Oracle Rdb versions: V7.1.4.4, V7.1.4.5, V7.2.0.1, V7.2.0.2 and V7.2.1. This problem has been corrected in Oracle Rdb Release 7.2.1.1.

6.1.9 ALTER Requires Read−Write Access to Storage Areas

In Oracle Rdb Release 7.2.1, an SQL ALTER TABLE, ALTER STORAGE MAP, TRUNCATE TABLE or ALTER INDEX statement that potentially modifies the on−disk row length requires that all partitions of the object reside in storage areas that allow read−write access. Further, the RDB$SYSTEM storage area is also required to allow read−write access.

The error "RDMS−F−READONLY, data in a read−only storage area may not be accessed for update" will be returned indicating that the storage area being referenced does not allow read−write access as in this example:

```
SQL>ALTER DATA FILE 'PLUGH' ALTER STORAGE AREA U2 READ ONLY;
    .
    .
    .
SQL>ALTER TABLE PARTUNIF ADD COLUMN RDB_TEST1 INT;
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−F−READONLY, data in a read−only storage area may
not be accessed for update
```

The restriction that the modified object reside in storage areas that allow read−write access has been lifted in this release, Oracle Rdb Release 7.2.1.1.

6.2 SQL Errors Fixed

6.2.1 SQL/Services Executor Loops Consuming 99% CPU

Bugs 4401924 and 5353228

After upgrading to SQL/Services 7.1.5.9.1 with Rdb 7.1.4.3.1 and later versions, some applications occasionally experienced a SQL/Services executor which enters a tight loop consuming a very high percentage of the CPU until it is stopped. This problem typically happens immediately after an executor begins servicing a new client and severely degrades the performance of everything else on the VMS node where it occurs. The only way to clear the problem is to STOP/ID the looping executor process.

The problem was caused by memory corruption associated with SQL's management of cached metadata. The memory corruption was associated with multiple SQL connections where some connection has cursors defined. It usually occurred after a DISCONNECT statement but might occur at other times. As such, it could affect any application which uses SQL connections and cursors, not just a SQL/Services executor as described above. Depending on what memory was corrupted, a wide variety of symptoms was also possible including failures in other layers of Rdb as well as in the customer application.

Note: A change was included in V7.1.2 which reduced the impact of this problem for customers experiencing the looping behavior. That change was not a fix.

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.2.1.1.

6.2.2 Some SQL Dialect−required Warnings Not Delivered

Bugs 3651847 and 4532451

The required warnings (information codes) for such things as rows eliminated for nulls (%RDB−I−ELIM_NULL) and string truncation (%RDB−I−TRUN_RTRV) were not being returned for singleton SELECT and singleton UPDATE statements; that is, statements that return a single row using the INTO clause. This could be demonstrated with a PERSONNEL database using the following interactive SQL commands.

```
SQL> set dialect 'sql92';
SQL> attach 'filename sql$database';
SQL>
SQL> ! Force a row to contain NULL for SALARY_AMOUNT
SQL> update salary_history
cont> set salary_amount = NULL
cont> where employee_id = '00471'
cont>   and salary_end = date vms'20−Aug−1981';
1 row updated
SQL>
SQL> declare :avg_sal integer(2);
SQL>
SQL> ! No informational generated (but is expected)
SQL> select avg(salary_amount) into :avg_sal
cont> from salary_history where employee_id = '00471'
```

```
cont>  and salary_end >= date vms'1-AUG-1970';
SQL> show sqlca
SQLCA:
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        0
        SQLERRD:        [0]: 0
                        [1]: 0
                        [2]: 1
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       00000
SQL> print :avg_sal;
        AVG_SAL
       60893.86
SQL> rollback;
```

The required SQLCODE and SQLSTATE values are now returned. In the example above, this causes the following differences:

```
SQL> select avg(salary_amount) into :avg_sal
cont> from salary_history where employee_id = '00471'
cont>  and salary_end >= date vms'1-AUG-1970';
%RDB-I-ELIM_NULL, null value eliminated in set function
SQL> show sqlca
SQLCA:
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        1003
        SQLERRD:        [0]: 0
                        [1]: 0
                        [2]: 1
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       01003
%RDB-I-ELIM_NULL, null value eliminated in set function
```

This problem has been corrected in Oracle Rdb Release 7.2.1.1.

6.2.3 SET AUTOMATIC TRANSLATION 'ON' May Cause Wrong Results From Queries

Bug 5849845

In prior releases of Oracle Rdb V7.2, the SET AUTOMATIC TRANSLATION 'ON' command could cause some queries to return the wrong results. This occurred when a text string literal, parameter or variable was compared with a numeric column. AUTOMATIC TRANSLATION was erroneously processing the non-text values.

The following example shows that the result is incorrect when the SET AUTOMATIC TRANSLATION 'ON' command is used in the session.

```
SQL> set automatic translation 'off';
SQL> select * from TEST_NLS_D where num = '333.333';
%SQL-I-NUMCMPTXT, Numeric column will be compared with string literal as text
             NUM   TDATE
         333.333    3-MAR-2099 00:00:00.00
1 row selected
SQL>
SQL> set automatic translation 'on';
SQL> select * from TEST_NLS_D where num = '333.333';
%SQL-I-NUMCMPTXT, Numeric column will be compared with string literal as text
0 rows selected
SQL> rollback;
```

This problem has been corrected in Oracle Rdb Release 7.2.1.1. The default is 'OFF' for most Oracle Rdb users. However, recent versions of OCI Services for Rdb use this command and therefore queries through OCI applications may be impacted.

6.2.4 Declared Variables Ignored by Oracle Dialect Dynamic Statements

Bug 5847260

In previous versions of Oracle Rdb, it was possible that dynamic statements would ignore variables created with the DECLARE syntax. This occurred when the dialect was set to ORACLE LEVEL1 or ORACLE LEVEL2.

This problem has been corrected in Oracle Rdb Release 7.2.1.1. The Oracle dialect now correctly identifies these as declared variables and not as parameter markers.

6.2.5 Unexpected DATTYPUNK Error Reported When Parameter Appeared in CONCAT Operation

Bug 5847260

In Oracle Rdb Release 7.2.1, an error was reported when processing dynamic SQL statements if a statement included a concatenation with a parameter marker. The following example shows this error:

```
-> UPDATE EMPLOYEES SET ADDRESS_DATA_2='My Address '  || :B1  WHERE
EMPLOYEE_ID = :B1;
Error -1:
%SQL-F-DATTYPUNK, Data type unknown. Expression cannot use only host variables
```

This problem has been corrected in Oracle Rdb Release 7.2.1.1. Concatenate will now assign the default VARCHAR(2000) type to the parameter so that the final result length for the string can be calculated. In prior versions, the parameter was defaulted to the length of the first parameter which may have been too short.

6.2.6 AIP Length Not Set by ALTER TABLE for Unmapped Tables

Oracle Rdb Release 7.2.1.0 added support for updating the length in the AIP. However, ALTER TABLE for

tables that do not have a STORAGE MAP with a STORE clause are not currently having the length updated in the AIP.

The following example shows the problem (no LOGMODVAL message in the log output).

```
SQL> set flags 'stomap_stats';
SQL>
SQL> create table T (a integer);
SQL>
SQL> alter table T
cont>     add column b varchar(230);
~As: reads: async 0 synch 8, writes: async 10 synch 0
SQL>
SQL> commit;
SQL>
SQL> create storage map T_MAP
cont>     for T
cont>     store in RDB$SYSTEM;
~As: create storage map "T_MAP"
~As: Table "T" (sys=0, rest=0, tmptbl=0)
~As: creating storage mapping routine T_MAP (columns=0)
~As: creating system module RDB$STORAGE_MAPS
SQL>
SQL> alter table T
cont>     add column c timestamp(2);
~As: reads: async 0 synch 15, writes: async 11 synch 0
SQL>
SQL> commit;
%RDMS-I-LOGMODVAL,     modified record length to 252
%RDMS-W-REBUILDSPAMS, SPAM pages should be rebuilt for logical area T
~As unlocking table "T" (PU -> PR)
```

To workaround this problem, you can add a storage map to the table as shown in the example above, or use the new RMU Set AIP command as shown in the following example.

```
SQL> set flags 'stomap_stats';
SQL>
SQL> create table T (a integer);
SQL>
SQL> alter table T
cont>     add column b varchar(230);
~As: reads: async 0 synch 8, writes: async 10 synch 0
SQL>
SQL> commit;
$ define/user rdms$set_flags "stomap_stats"
$ rmu/set aip abc t/length
%RDMS-I-LOGMODVAL,     modified record length to 244
%RDMS-W-REBUILDSPAMS, SPAM pages should be rebuilt for logical area T
```

This problem has been corrected in Oracle Rdb Release 7.2.1.1.

6.3 RMU Errors Fixed

6.3.1 RMU Online Verification ACCVIO at RMUVLAREA$VERIFY_LAREA_PAGES

During an online RMU /VERIFY operation, if a storage area extended while the verification was running, it was possible for an internal data structure to be undersized. This could lead to several possible failure cases including an RMU bugcheck dump file with a footprint similar to the following.

```
***** Exception at 000000000080B101 :
    RMU72\RMUVLAREA$VERIFY_LAREA_PAGES + 00000581
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=00000000014E0005, PC=000000000080B101, PS=0000001B
Saved PC = 000000000080A6B0 : RMU72\RMUVLAREA$VERIFY_ALL_LAREAS + 00000390
Saved PC = 00000000007BA110 : RMU72\RMUVER$VERIFY + 00003100
Saved PC = 0000000000470820 : RMU72\RMU$VERIFY + 00002E70
Saved PC = 0000000000463A30 : RMU72\RMU_DISPATCH + 00002840
Saved PC = 0000000000460A50 : RMU72\RMU_STARTUP + 00000910
```

Generally, running the verify operation again would complete correctly.

This problem has been corrected in Oracle Rdb Release 7.2.1.1. Access to the data structure now avoids access outside of the allocated memory.

6.3.2 RMU/SHOW LOGICAL_NAMES Does Not Include RDM$MONITORnn

Bug 5847856

Previously, the list of logical names displayed by the RMU/SHOW LOGICAL_NAMES command did not include the logical name "RDM$MONITORnn" though it did include the logical name "RDM$MONITOR".

This problem has been corrected in Oracle Rdb Release 7.2.1.1. The logical name "RDM$MONITORnn" (where "nn" refers to the multi−version Rdb installation version of RMU being executed) is now displayed.

6.3.3 RMU/VERIFY/INCREMENTAL Incorrect Diagnostics for Bitmap Indexes

Bug 4149656

Even though RMU/VERIFY knew it was walking a btree bitmap index, a bad flag passed to a routine verifying the index could cause "hashed" to be incorrectly inserted in the message for the index type. Also, invalid bitmap btree index diagnostics such as RMU−W−BADHDADBK, RMU−I−BTRDUPCAR, and RMU−I−BTRERPATH could be output by RMU/VERIFY/INCREMENTAL even though the index did not have any problems. This was caused by a loss of context while RMU/VERIFY was validating the btree index duplicate bitmap chain.

This problem only happened when RMU/VERIFY/INCREMENTAL was verifying btree bitmap index duplicate chains. It did not happen if /NODATA was specified. It happened on longer bitmap chains that required fetching multiple duplicate bitmap records.

The following example shows invalid index diagnostics output by RMU/VERIFY/INCREMENTAL.

```
$ RMU/VERIFY/INCREMENTAL/ALL TEST_DATABASE
%RMU-W-BADHDADBK, Bad logical area DBID in hashed index data record
dbkey 1716:674953227:-32678 Found 06B4 (hex).
%RMU-W-BADHDADBK, Bad logical area DBID in hashed index data record dbkey
1716:674953227:-32677 Found 06B4 (hex).
%RMU-W-BADHDADBK, Bad logical area DBID in hashed index data record dbkey
1716:674953227:-32676 Found 06B4 (hex).
```

A workaround for this problem is to either specify /NODATA or to not specify /INCREMENTAL for the verify.

```
$ RMU/VERIFY/INCREMENTAL/INDEX/NODATA TEST_DATABASE
$ RMU/VERIFY/ALL TEST_DATABASE
```

This problem has been corrected in Oracle Rdb Release 7.2.1.1.

6.3.4 RMU/BACKUP With PARALLEL and COMPRESS=ZLIB Fails

Bug 5870330

Executing an RMU/BACKUP command with PARALLEL and COMPRESS=ZLIB qualifiers caused a bugcheck with the following footprint:

```
Alpha OpenVMS 8.2
Oracle Rdb Server 7.2.1.0.1
Got a RMUBUGCHK.DMP
SYSTEM-F-ACCVIO, access violation, virtual address=0000000000000000
Exception occurred at RMUSHR72\COSIZLIB$FINISH_COMPRESS + 00000048
Called from RMUSHR72\RMUBCK$BF_BACKUP_THREAD + 00001244
Called from RMUSHR72\RMUIO$TERMINATE_THREAD + 00000040
Called from RMUSHR72\RMUIO$FIREWALL + 00000040
Running image RMUEXEC72.EXE
Dump created: 7-FEB-2007 08:45:58.93
```

An oversight in the backup code used an uninitialized compression context structure. The same error could happen without using PARALLEL but using COMPRESS=ZLIB and having more than one tape drive for output.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.2.1.1.

6.4 Row Cache Errors Fixed

6.4.1 Bugcheck During Online RMU Backup When Snapshots In Row Cache Enabled

Bug 4260102

In rare situations when using the "snapshots in row cache" feature, a cached database row may be modified to become larger. This modification may require allocation of space on the database page. In this case, it is possible that a row snapshot is not written to the database but rather maintained solely in the row cache. An online RMU operation (such as RMU /BACKUP) may be unable to locate the snapshot row copy and may then fail with a BADPAGNUM, PAGE 4294967295 error in the dump file.

This problem has been corrected in Oracle Rdb Release 7.2.1.1. The snapshot chain for the row that is having its size adjusted is written to the snapshot storage area before the actual on−disk row modification is made.

6.4.2 Slight Relaxation of VMS$MEM_RESIDENT_USER Requirement

Bug 5859487

Previously, the VMS$MEM_RESIDENT_USER identifier was required to open a database that had any row cache configured for resident memory even if no caches were enabled for the database.

This restriction has been relaxed in Oracle Rdb Release 7.2.1.1. If the database is not enabled for row caches, the VMS$MEM_RESIDENT_USER identifier is not required even if caches are defined for resident memory.

6.5 RMU Show Statistics Errors Fixed

6.5.1 Latch Hangs Possible From RMU /SHOW STATISTICS

Bugs 4397634 and 5842040

In prior releases of Oracle Rdb, it was possible in a very small timing window for processes running the RMU /SHOW STATISTICS command to become hung while manipulating "latches" during the database attach sequence. Depending on the exact timing and sequence of events, this process might block other users of the database.

This problem has been corrected in Oracle Rdb Release 7.2.1.1.

6.5.2 RMU/SHOW STATISTICS Bugcheck in KUTDIS$LONG_TX_NOTIFY

Bug 5867253

In Oracle Rdb Release 7.2.1, it was possible for the RMU /SHOW STATISTICS command to fail with a bugcheck dump when using the configuration option LONG_TX_SECONDS. The bugcheck dump footprint would be similar to the following:

```
SYSTEM-F-ACCVIO, access violation
Exception occurred at RMU72\KUTDIS$LONG_TX_NOTIFY + 00000424
Called from RMU72\KUTDIS$EVENT_NOTIFY +  00000054
Called from RMU72\KUTDIS$DISPLAY_ASTX + 00000584
Called from RMU72\KUT$DISPLAY + 0000199C
Running image RMU72.EXE
Command line was "RMUI/SHOW STATISTIC
DSA0:[DB]FOO.RDB;/NOINTERACTIVE/CONFIGURATION=DSA0:[FOO]FOO.CFGCFG"
```

This problem has been corrected in Oracle Rdb Release 7.2.1.1.

6.6 Hot Standby Errors Fixed

6.6.1 LRS Shutdown Failure RDMS−F−PARTDTXNERR/SYSTEM−F−NOSUCHID

Bug 5754461

A possible problem with the Oracle Rdb Hot Standby feature has been identified. If the OpenVMS $GETGTI system service returns a status value of SS$_NOSUCHID, the LRS process might be unable to shutdown cleanly. This could result in an inconsistent standby database.

This problem has been corrected in Oracle Rdb Release 7.2.1.1. The LRS process now treats a returned SS$_NOSUCHID status the same as a SS$_NOSUCHTID status and it will be handled normally and will not cause the LRS to fail.

Chapter 7
Software Errors Fixed in Oracle Rdb Release 7.2.1.0

This chapter describes software errors that are fixed by Oracle Rdb Release 7.2.1.0.

7.1 Software Errors Fixed That Apply to All Interfaces

7.1.1 Incorrect Backup Checksum and CRC Values on I64

In some cases, the checksum or CRC values within an .RBF backup file on I64 systems starting with Rdb Release V7.2.0.2 may be incorrect. This difference could result in checksum errors during restore operations when using /CRC=CHECKSUM. In other words, the restore on Rdb 7.2.0.2 I64 cannot read backups made by any other platform or version when CRC=CHECKSUM was used.

As a workaround, Oracle recommends using the default CRC algorithm of /CRC=AUTODIN_II rather than /CRC=CHECKSUM.

This problem has been corrected in Oracle Rdb Release 7.2.1. The checksum value calculated by Oracle Rdb is now the same on all platforms and versions.

7.1.2 Bugcheck Loop Created Many Bugcheck Dumps

Bugs 5411895 and 5361954

In some rare circumstances, a bugcheck dump could cause another bugcheck dump to occur, resulting in what would be an infinite number of bugcheck dumps, except that a finite number would, in fact, be produced. The limit to the number of bugcheck dumps being created was due to using up available disk space.

An attempt to remedy this issue has been made. After a process has created three bugcheck dumps, any further dumps should become "mini" dumps. After three mini bugcheck dumps, the dumps should cease and a COSI$_FATINTERR status returned.

All further bugcheck dump attempts should simply return COSI$_FATINTERR (in other words, no more bugcheck dumps will be produced).

An example of a query causing this problem is shown below.

```
SELECT   TIPO, OPFU
FROM     T1
WHERE    CODE  ='100' AND
         VALUE = '0057984193004785667' AND
         COOP  = 'OCA0604WOREST' AND
         CRTC = 'MFV' AND
         OPFU  = '20011228' AND
       ( TIPO  = 'TCI' OR
         (IVSN = 'I' AND (TIPO = 'COM' OR TIPO  = 'ATR') ) ) ;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
...etc...
```

This change has been made in Oracle Rdb Release 7.2.1.

7.1.3 Left Outer Join Query Slows With Full Index Scan

Bug 5567495

A customer reported that a particular query runs significantly slower after upgrading from Release 7.1.4.1 to Release 7.1.4.4 when using full index scan in the outer leg of the Left Outer join operation. We were able to reproduce the problem using the following simple script without data, since the difference in the output strategy is indicative of the problem.

```
create table t1 (
    col_one integer,
    col_two integer,
    price   integer);

create table t2 (
    col_one     integer,
    col_two     integer,
    line_amt    COMPUTED BY
    (select c1.price from t1 c1
     where ((c1.col_one = t2.col_one)
         and (c1.col_two = t2.col_two)))) ;

create unique index t1_ndx on t1 (col_one, col_two);
create unique index t2_ndx on t2 (col_one, col_two);

create view v_t1_loj_t2
    (col_one, col_two, total) as
    (select c2.col_one, c2.col_two, SUM(c3.line_amt)
    from t1 as c2
       LEFT OUTER JOIN
         t2 as c3
            ON ((c3.col_one = c2.col_one) AND
  (c3.col_two = c2.col_two))
    GROUP BY c2.col_one, c2.col_two);

! The following is the query that slows down in performance since the
! strategy applies a full index scan at the outer leg of the left outer join
! operation, as compared to "Direct lookup" index retrieval strategy.

select t1.col_two,
    (select v1.total from v_t1_loj_v2 v1
     where v1.col_one = t1.col_one and
            v1.col_two = t1.col_two) as v_total
from t1 where t1.col_one = 1 ;
Cross block of 2 entries
  Cross block entry 1
    Conjunct        Index only retrieval of relation T1
      Index name  T1_NDX [1:1]
  Cross block entry 2
    Aggregate       Conjunct        Aggregate
    Cross block of 2 entries
      Cross block entry 1
        Match    (Left Outer Join)
          Outer loop
            Index only retrieval of relation T1
              Index name  T1_NDX [0:0]               <== Full index scan
          Inner loop      (zig-zag)
            Index only retrieval of relation T2
              Index name  T2_NDX [0:0]
      Cross block entry 2
```

```
        Aggregate       Get     Retrieval by index of relation T1
          Index name  T1_NDX [2:2]   Direct lookup
0 rows selected
```

The strategy from Rdb Release 7.1.4.1.1 is exactly the same with the exception of the better performing index retrieval in the Outer loop of the Left Outer Join operation.

```
Cross block of 2 entries
  Cross block entry 1
    Conjunct        Index only retrieval of relation T1
      Index name  T1_NDX [1:1]
  Cross block entry 2
    Aggregate       Conjunct        Aggregate
    Cross block of 2 entries
      Cross block entry 1
        Match    (Left Outer Join)
          Outer loop
            Index only retrieval of relation T1
              Index name  T1_NDX [2:2]   Direct lookup      <== Best one
          Inner loop      (zig-zag)
            Index only retrieval of relation T2
              Index name  T2_NDX1 [0:0]
      Cross block entry 2
        Aggregate       Get     Retrieval by index of relation T1
          Index name  T1_NDX [2:2]   Direct lookup
```

There is no known 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 selects from a table (T1) using the WHERE filter predicate on the column which is used as one of the join items in the subselect query to join another view.
2. The view is defined as an aggregate left outer join query between T1 and a second table (T2) on the two join columns.
3. The last column of the view is an aggregate function SUM on the COMPUTED BY column of the table T2 which contains a SELECT query to join table T1 using the same two columns.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.1.4 DBR Bugchecks at DIO$LACB_CREATE + 00000174

Bug 5467328

It was possible for the database recovery process (DBR) to bugcheck when a process terminated while rolling back the creation of a new table or index. For example, consider the following script:

```
SQL> CREATE DATABASE FILENAME TEST;
SQL> CREATE TABLE T1 (C1 INT);
SQL> INSERT INTO T1 VALUES (1);
1 row inserted
SQL> ROLLBACK;
```

In the above sequence of commands, if during the course of rolling back the transaction, the process were to rollback the creation of table T1, update the area−inventory page (AIP) for the table, and flush the changed AIP entry to disk, but get terminated before it could truncate the recovery−unit journal (RUJ), then the DBR

would fail when attempting to access the non−existant table T1. The first DBR bugcheck would contain an exception similar to the following:

```
***** Exception at 00000000001A6C0C : RDMDBR72\DIOLACB$LACB_AIP_ENT_GET +
000005CC
%COSI−F−BUGCHECK, internal consistency failure
```

Subsequent attempts to access the database would result in a DBR bugcheck with an exception similar to the following:

```
***** Exception at 00000000001A6174 : RDMDBR72\DIO$LACB_CREATE + 00000174
%RDMS−F−CANTFINDLAREA, cannot locate logical area 58 in area inventory page list
```

This problem would occur because the DBR was not prepared for the possibility that a table might not exist when it attempted to rollback inserts for the table.

If this problem is encountered, there is no supported workaround to resolve the problem. The database must be restored and recovered or Oracle Rdb must be updated to the release that contains the fix for this problem before attempting to access the database.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.1.5 Processes Do Not Always Terminate After Monitor Terminates

Bug 5361981

When the Oracle Rdb monitor process terminates abnormally, all user processes that are attached to databases on that node should immediately terminate. However, there were cases where that didn't happen, and those user processes would continue to access Oracle Rdb resources after the monitor failed. Consider the following example.

1. User 1, node 1: *SQL> ATTACH 'FILENAME MF_PERSONNEL';*
2. User 2, node 2: *SQL> ATTACH 'FILENAME MF_PERSONNEL';*
3. User 3, node 1: *$ STOP/ID={pid of monitor process on node 1}*

In the above sequence of events, the user process on node 1 should have terminated as soon as the monitor process was killed, but it remained active.

This problem can be avoided by using the RMU/OPEN command and manually opening databases on all nodes that will have users accessing the database.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.1.6 Wrong Results With "OR Index Retrieval" and Bitmapped Scan

Bug 5363170

When a strategy was chosen by the optimizer that included "OR Index retrieval" and bitmapped scan was enabled, queries could return incorrect results.

The following example should return 100 rows, but only returns one. Note that bitmap scan is enabled and the strategy includes "OR index retrieval" on the second (inner) cross block.

```
SQL> set flags 'bitmap,strategy,detail'
SQL> select t1.id from t1 join t2
cont>  on t1.id=t2.id or t1.id=t2.id
cont>  where t1.f2=1 and t1.f2=1;
Tables:
  0 = T1
  1 = T2
Cross block of 2 entries
  Cross block entry 1
    Conjunct: 0.F2 = 1
    Conjunct: 0.F2 = 1
    Get     Retrieval sequentially of relation 0:T1
  Cross block entry 2
    Conjunct: ((0.ID = 1.ID) OR (0.ID = 1.ID)) AND (0.F2 = 1) AND (0.F2 = 1)
               AND (0.ID = 1.ID)
    OR index retrieval
      Index only retrieval of relation 1:T2
        Index name  I2 [1:1]
          Keys: 0.ID = 1.ID
 T1.ID
 00164
1 row selected
```

The problem can be avoided by not using bitmap scan.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.1.7 Bugcheck in Query on a Single Table With AND/OR Predicates

Bug 5361954

The customer states that a specific query always bugchecks with the following query after migrating a database from Alpha Rdb 7.0.6.5 to Itanium 7.2.0.0.

```
SELECT   TIPO, OPFU
FROM     T1
WHERE    CODE  ='100' AND
         VALUE = '0057984193004785667' AND
         COOP  = 'OCA0604WOREST' AND
         CRTC = 'MFV' AND
         OPFU  = '20011228' AND
       ( TIPO  = 'TCI' OR (IVSN = 'I' AND (TIPO = 'COM' OR TIPO  = 'ATR') ) ) ;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
%RDB-F-BUG_CHECK, internal consistency check failed
```

The problem is not specific to the Rdb 7.2 Integrity platform. The bugchecks have been seen with all Rdb versions.

Sometimes the query no longer fails if the query is executed after a metadata query such as "select * from rdb$database". Performing an EXPORT/IMPORT also does not show the problem.

The query works if the SQL flags 'MAX_STABILITY' is defined, as in the following example.

```
SET FLAGS 'MAX_STABILITY';

SELECT   TIPO, OPFU
FROM     T1
WHERE    CODE  ='100' AND
         VALUE = '0057984193004785667' AND
         COOP  = 'OCA0604WOREST' AND
         CRTC  = 'MFV' AND
         OPFU  = '20011228' AND
       ( TIPO  = 'TCI' OR (IVSN = 'I' AND (TIPO = 'COM' OR TIPO  = 'ATR') ) ) ;
Firstn                    Sort                       Conjunct
Get                       Retrieval by index of relation T1
  Index name  T1_NDX3 [5:5,(7:7)2] Bool
 TIPO   OPFU
 COM    20011228
1 row selected
```

The following indices must be defined for the table T1 for the query to fail.

```
T1_NDX1 with column CODE
    and column NUMOP

T1_NDX2 with column TIPO
    and column CODE
    and column VALUE
    and column COOP
    and column CRTC
    and column OPFU
    and column IVSN
    and column NUMOP

T1_NDX3 with column CODE descending
    and column VALUE descending
    and column COOP
    and column CRTC
    and column TIPO
    and column IVSN
    and column OPFU
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.1.8 Bugcheck When Bitmap Scan Enabled

Bug 5414635

The following query bugchecks when the bitmap scan feature is enabled.

```
set flags 'bitmap';
select * from T1 where C2 = 'XYZ' or C4 >='03-jul-2006' ;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file RDSBUGCHK.DMP;
```

The query works if the bitmap scan is disabled, as in the following example.

```
set flags 'nobitmap';
SQL> select * from T1 where C2 = 'XYZ' or C4 >='03-jul-2006' ;
Tables:
  0 = T1
Conjunct: (0.C2 = 'XYZ') OR (0.C4 >= '3-JUL-2006')
Get     Retrieval by index of relation 0:T1
  Index name  X1_T1 [0:0]
0 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.1.9 Wrong Result From Star Join Query With Aggregate

Bug 5192800

The following query looks like a star join between the table t0 and two other tables (t1 and t2). It should return zero rows, but instead it incorrectly finds one row.

```
select t0.acct_num, t1.scard_id
  from
    subscr t0
    inner join
    scard t1
    on
        (t1.acct_num = t0.acct_num
         and exists
           (select *
             from hwatt t3 where t3.acct_num = t1.acct_num))
    left outer join
    service t2
    on
      t2.acct_num = t0.acct_num and
      t2.dt_tim_stmp =
            (select max(t4.dt_tim_stmp)
              from service t4
              where t4.acct_num = t0.acct_num);
Tables:
  0 = SUBSCR
  1 = SCARD
  2 = SERVICE
  3 = HWATT
  4 = SERVICE
Cross block of 2 entries
  Cross block entry 1
    Cross block of 2 entries        (Left Outer Join)
      Cross block entry 1
        Cross block of 3 entries
          Cross block entry 1
            Get     Retrieval sequentially of relation 0:SUBSCR
          Cross block entry 2
            Aggregate: 0:MAX (4.DT_TIM_STMP)
            Conjunct: 4.ACCT_NUM = 0.ACCT_NUM
            Get     Retrieval sequentially of relation 4:SERVICE
          Cross block entry 3
            Conjunct: 1.ACCT_NUM = 0.ACCT_NUM
            Get     Retrieval sequentially of relation 1:SCARD
      Cross block entry 2
        Conjunct: (2.ACCT_NUM = 0.ACCT_NUM) AND (2.DT_TIM_STMP = <agg0>)
        Get     Retrieval sequentially of relation 2:SERVICE
```

```
  Cross block entry 2
    Aggregate-F1: 1:COUNT-ANY (<subselect>)
    Conjunct: 3.ACCT_NUM = 1.ACCT_NUM
    Get     Retrieval sequentially of relation 3:HWATT
 T0.ACCT_NUM   T1.SCARD_ID
           1            2
1 row selected
```

The query works if the equality predicate with the MAX aggregate subselect query is converted into a WHERE clause, as in the following example.

```
select t0.acct_num, t1.scard_id
  from
    subscr t0
    inner join
    scard t1
    on
        (t1.acct_num = t0.acct_num
         and exists
           (select *
             from hwatt t3 where t3.acct_num = t1.acct_num))
    left outer join
    service t2
    on
      t2.acct_num = t0.acct_num
where ! <== converted into WHERE clause
      h.dt_tim_stmp =
            (select max(sv2.dt_tim_stmp)
              from service sv2 where sv2.acct_num = ss.acct_num);
Tables:
  0 = SUBSCR
  1 = SCARD
  2 = SERVICE
  3 = HWATT
  4 = SERVICE
Cross block of 2 entries
  Cross block entry 1
    Cross block of 2 entries        (Left Outer Join)
      Cross block entry 1
        Cross block of 3 entries
          Cross block entry 1
            Get     Retrieval sequentially of relation 0:SUBSCR
          Cross block entry 2
            Conjunct: 1.ACCT_NUM = 0.ACCT_NUM
            Get     Retrieval sequentially of relation 1:SCARD
          Cross block entry 3
            Conjunct: <agg0> <> 0                  <== See Note
            Aggregate-F1: 0:COUNT-ANY (<subselect>)
            Conjunct: 3.ACCT_NUM = 1.ACCT_NUM
            Get     Retrieval sequentially of relation 3:HWATT
      Cross block entry 2
        Conjunct: 2.ACCT_NUM = 0.ACCT_NUM
        Get     Retrieval sequentially of relation 2:SERVICE
  Cross block entry 2
    Conjunct: 2.DT_TIM_STMP = agg1
    Aggregate: 1:MAX (4.DT_TIM_STMP)
    Conjunct: 4.ACCT_NUM = 0.ACCT_NUM
    Get     Retrieval sequentially of relation 4:SERVICE
0 rows selected
```

Note:: The conjunct "<agg0> <> 0" appears in the good strategy output at the top of the Aggregate for EXISTS statement, but this conjunct is missing in the strategy output of the problem query.

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

1. The main query looks like a star join, joining a main table (t0) to multiple tables (in this case, two tables t1 and t2).
2. Table t0 and t1 are inner joined followed by an EXISTS subselect query referencing t3.
3. Table t0 and t2 are left outer joined followed by an equality predicate with a MAX aggregate subselect query which joins table t0 and t4.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.1.10 Restriction Relaxed for Executing External Routines via Privileged Images

Bugs 4595983 and 4606819

In prior versions of Oracle Rdb, an external routine could not be activated if it was called from an image that had more privileges than the current process was granted by OpenVMS. This security precaution prevented an unsafe sharable image being substituted in the production environment.

The following example shows the error reported by Oracle Rdb.

```
%RDB-E-EXTFUN_FAIL, external routine failed to compile or execute successfully
-RDMS-E-RTNUSENOTALL, routine "GET_ONE" can not be used, too many privileges
available
```

This restriction required SQL/Services applications to define external routines with BIND ON SERVER to allow the routine to be activated outside the privileged environment. This also causes problems for RMU Load (when constraints, triggers and AUTOMATIC INSERT columns call external routines), RMU Unload (when COMPUTED BY columns and views call external routines), and RMU Verify (which evaluated constraints).

Starting with Rdb V7.2, you can use the INSTALL ADD/SHARE command to make the external routine's sharable image "known". The sharable image must be activated by system wide logical names with the /EXECUTIVE mode option. Such images are assumed to be trusted in the case where a privileged image (such as RMU) executes the external routine.

In the following example, assume there is a function named GET_ONE in a shared image called FUNCS.EXE which is built from a single module named FUNCS.OBJ. The database has the following external function definition:

```
SQL> create function get_one() returns integer;
cont> external
cont>  name RETURN_ONE
cont>  location 'MY_DIR:FUNCS.EXE'
cont>  language general
cont>  parameter style general
cont>  deterministic
cont>  comment is 'Always returns 1';
```

Suppose there is a SQL$PRE/CC application named "PRIV_APP" which must be installed with extra privileges and which contains the following code:

```
EXEC SQL select get_one() into :result from rdb$database;
printf("%d is the loneliest number\n");
```

If FUNCS.EXE is not installed and PRIV_APP is executed from an account which has lower privileges than those given PRIV_APP when it was installed, the call to GET_ONE() shown above will fail with a RTNUSENOTALL error. In order to be able to execute GET_ONE() from within PRIV_APP, locate the sharable image in SYS$SHARE, and install FUNCS.EXE using DCL similar to the following.

```
$ LINK/SHARE/NOTRACE/EXE=SYS$COMMON:[SYSLIB]FUNCS.EXE FUNCS,SYS$INPUT:/OPTIONS
SYMBOL_VECTOR=(RETURN_ONE=PROCEDURE)
$ SET FILE /PROTECTION=W:RE SYS$SHARE:FUNCS.EXE
$ INSTALL ADD SYS$SHARE:FUNCS.EXE /SHARE
$ DEFINE/SYSTEM/EXECUTIVE_MODE MY_DIR SYS$SHARE
```

Using the installed sharable image PRIV_APP will now successfully execute the SQL statement that results in a successful call to the function GET_ONE.

This problem was corrected in Oracle Rdb Release 7.2.

7.1.11 RDMS$_ Symbols Missing From RDMMSGSHR on I64

Bug 5309253

In prior Oracle Rdb 7.2 releases on I64, all message symbols were erroneously omitted from RDMMSGSHR.EXE.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.1.12 Hangs or Looping When Lots of Page Contention

Applications that had lots of page contention could sometimes hang due to page locks not being released by a process or they could enter a CPU loop. This problem was only in Oracle Rdb Release 7.2.

This problem would occur when an internal queue used to manage blocking AST requests would become corrupt. In that situation, blocking ASTs could be lost or processing of the queue could result in an infinite loop.

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.1.13 Logical RDMS$ENABLE_INDEX_COLUMN_GROUP Not Working

Bug 5614198

Users are unable to disable this feature in Oracle Rdb Release 7.2 using the VMS logical RDMS$ENABLE_INDEX_COLUMN_GROUP.

```
$define RDMS$ENABLE_INDEX_COLUMN_GROUP 0
$SQL$
attach 'file personnel';
show flags;
Alias RDB$DBHANDLE:
Flags currently set for Oracle Rdb:
   STRATEGY,PREFIX,WARN_DDL,INDEX_COLUMN_GROUP,MAX_SOLUTION
   ,MAX_RECURSION(100),DETAIL_LEVEL(1),REFINE_ESTIMATES(127)
   ,NOBITMAPPED_SCAN
```

The workaround is to use the command SQL FLAGS 'NOINDEX_COLUMN_GROUP'.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.2 SQL Errors Fixed

7.2.1 CREATE OUTLINE Not Fully Supported for MULTISCHEMA Databases

Well formed query outlines do not always work correctly within a multischema database. This is because the object name is used instead of the STORED NAME for the MODULE, RELATION and INDEX tags. The only time these references work is when the STORED NAME is the same as the name within the schema.

The following example shows the errors reported by Oracle Rdb due to the incorrectly passed names.

```
SQL> create outline S.QO_0
cont> id 'DAE28B9C6DA276E600C68C32AFF46F88'
cont> mode 0
cont> as (
cont>   query (
cont> -- Select
cont>     subquery (
cont>       TT      MODULE S.M2 0     access path sequential
cont>       )
cont>     )
cont>   )
cont> compliance optional     ;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-E-MODNEXTS, module M2 does not exist in this database
SQL>
SQL> create outline S.QO
cont>     stored name is "qoQOqo"
cont> id '74263C5C965F88554C9E67744616925C'
cont> mode 0
cont> as (
cont>   query (
cont> -- For loop
cont>     subquery (
cont>       S.TTABLE 0      access path sequential
cont>       )
cont>     )
cont>   )
cont> compliance optional     ;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-RELNOEXI, relation TTABLE does not exist in this database
SQL>
SQL> create outline S.QO_2
cont> id '21CA5C0637609367779EB2D7967FF11B'
cont> mode 0
cont> as (
cont>   query (
cont> -- For loop
cont>     subquery (
cont>       S.T 0     access path index       S.T_INDEX
cont>       )
cont>     )
cont>   )
cont> compliance optional     ;
```

```
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-F-INDNOTEXI, index T_INDEX does not exist in this database
```

SQL was also not processing the declared local temporary (aka scratch) table name correctly. The CREATE OUTLINE syntax allows the schema (and catalog) for the table name but in the case of a temporary table within a module, the table sub-object is fully specified by the module name. SQL now restricts the name to be unqualified.

These problems have been corrected in Oracle Rdb Release 7.2.1.

7.2.2 Unexpected INV_TBL_DCL Error From CREATE MODULE in Compiled Source

If either a SQL Module Language or SQL Precompiler source file contained a CREATE MODULE statement that used DECLARE LOCAL TEMPORARY TABLE, several errors were reported. The same CREATE MODULE statement was acceptable in Interactive or Dynamic SQL.

The following example shows these errors.

```
    declare local temporary table module.T2T (f float)
                                          1
%SQL-E-INV_TBL_DCL, (1) Invalid use of declared local temporary table T2T
    select f into :x from module.T2T where module.T2T.f = 0e0;
                          1
%SQL-F-RELNOTDCL, (1) Table T2T has not been declared in module or environment
```

This restriction has been lifted with this release of Oracle Rdb. The prohibition was being incorrectly applied to CREATE MODULE statements and the restriction on DECLARE LOCAL TEMPORARY TABLE does not apply to DDL statements in a compiled module.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.2.3 Numeric Out of Range SQLSTATE 22003 Not Returned

Bug 5208860

For a table column defined as NUMERIC, if an out-of-range value were inserted in the column, the returned SQLCODE would be -1 with a SQLSTATE of RR000 in lieu of the proper SQLCODE of -304 and SQLSTATE of 22003.

For example, consider a table defined as follows:

```
CREATE TABLE NUM1 ( NUM1C1 NUMERIC (3, 2),  NUM1C2 NUMERIC (2) );
```

The following example illustrates the old, incorrect SQLCODE and SQLSTATE returned when an out-of-range value is inserted into the table:

```
SQL> INSERT INTO NUM1 VALUES (-10, 0);
%RDB-E-VALOUTRANGE, value outside the specified precision (3) for column
"NUM1C1"
SQL> show sqlca
SQLCA:
```

```
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        -1
        SQLERRD:        [0]: 2
                        [1]: 0
                        [2]: 0
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       RR000
```

This INSERT now produces the following results:

```
SQL> INSERT INTO NUM1 VALUES (-10, 0);
%RDB-E-VALOUTRANGE, value outside the specified precision (3) for column
"NUM1C1"
SQL> show sqlca
SQLCA:
        SQLCAID:        SQLCA           SQLCABC:        128
        SQLCODE:        -304
        SQLERRD:        [0]: 0
                        [1]: 0
                        [2]: 0
                        [3]: 0
                        [4]: 0
                        [5]: 0
        SQLWARN0:       0       SQLWARN1:       0       SQLWARN2:       0
        SQLWARN3:       0       SQLWARN4:       0       SQLWARN5:       0
        SQLWARN6:       0       SQLWARN7:       0
        SQLSTATE:       22003
```

There is no known workaround to get the correct SQLCODE/SQLSTATE.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.2.4 TIMESTAMP Result of DATE Added to TIME Expression was Truncated

Oracle Rdb allows the addition of DATE ANSI and a TIME data type value to produce a TIMESTAMP result. SQL incorrectly assigned a fractional seconds precision to the TIMESTAMP result of zero (0). Therefore, the result was truncated by Interactive SQL. SQL now correctly assigns the fractional seconds precision of the TIME value expression to the TIMESTAMP result.

The following example shows that the fractional seconds were truncated.

```
SQL> select date ansi'2006-1-1' + time'12:12:12.12' from rdb$database;

 2006-01-01 12:12:12
1 row selected
```

This problem has been corrected in Oracle Rdb Release 7.2.1. SQL now correctly assigns the fractional seconds precision of the TIME value expression to the TIMESTAMP result.

7.2.5 Unexpected Constraint Failure from INSERT ... SELECT Statement

In prior releases of Oracle Rdb, NOT DEFERRABLE constraints were evaluated for each row inserted by the INSERT INTO ... SELECT ... FROM ... statement. In some cases, this row by row evaluation might cause the INSERT statement to fail when it was expected to succeed. The ANSI and ISO Database Language standard for SQL specifies that the INSERT statement from a SELECT is atomic and constraint evaluation should be performed after all rows are inserted.

The following example shows a constraint that fails in a case where it should have succeeded.

```
SQL> set dialect 'SQL99';
SQL>
SQL> create table TEST_A (a integer);
SQL> insert into TEST_A values (1);
1 row inserted
SQL> insert into TEST_A values (-1);
1 row inserted
SQL>
SQL> create table TEST_B
cont>     (b integer);
SQL>
SQL> insert into TEST_B select * from TEST_A;
2 rows inserted
SQL>
SQL> -- add constraint that ensures total is zero
SQL> alter table TEST_B
cont>     add constraint BB
cont>   check ((select sum (b) from TEST_B) = 0)
cont>   not deferrable;
SQL>
SQL> insert into TEST_B select * from TEST_A;
%RDB-E-INTEG_FAIL, violation of constraint BB caused operation to fail
-RDB-F-ON_DB, on database DISK1:[DATABASES]MF_PERSONNEL.RDB;1
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.1. If the SQL language dialect SQL92, SQL99, ORACLE LEVEL1 or ORACLE LEVEL2 is set, the NOT DEFERRABLE constraint evaluation is now performed after all rows have been inserted for the INSERT ... SELECT statement.

7.2.6 SQL$MOD /LIS Displays Incorrect /ALIGN Value

Bug 5350528

In some cases, the listing produced by the SQL Module Language compiler would display an incorrect value for the /(NO)ALIGN qualifier used in the command line summary section at the end of the listing. The correct value of the qualifier was used in the compilation; the problem was restricted to a misleading listing.

For example, the following might appear in the command line summary section of the listing:

```
Command Line Summary:

    /LIST/NOALIGN/MACH TEST$SOURCE:MOD_DATETIME_ADA_4.SQLMOD
```

```
/FLOAT=D_FLOAT
/WARNING=(WARNING, DEPRECATED)
/NOFLAG_NONSTANDARD
/CONSTRAINT_MODE=DEFERRED
/NOCONNECT
/INITIALIZE_HANDLES
/NORESTRICT_INVOKER
/ALIGN_RECORDS
    ...
```

In the above example, the command line specified /NOALIGN but the value shown as being used is /ALIGN_RECORDS even though the records were actually not aligned.

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.2.7 FOR UPDATE Clause was Ignored for DECLARE CURSOR

Bug 4990176

Starting with Oracle Rdb Release 7.1.2, the SELECT ... FOR UPDATE clause was considered synonymous with the UPDATE ONLY CURSOR syntax. The FOR UPDATE clause is used by various SQL dialects to imply stronger locking during the initial read of a row. For instance, a singleton SELECT statement might use FOR UPDATE to fetch the row and save the DBKEY (or ROWID) for subsequent updating by the application.

For Dynamic and Interactive SQL, the FOR UPDATE is an important feature when using cursors that update rows. In these environments, the intent to update the cursor is not known until the UPDATE ... WHERE CURRENT OF statement is encountered. The FOR UPDATE clause not only provides valuable information for the optimization of the query but also locks the rows ready for update. However, the DECLARE CURSOR syntax was ignoring the FOR UPDATE clause in all dialects except ORACLE LEVEL1 and ORACLE LEVEL2 and no strong locking was applied by Oracle Rdb.

This problem has been corrected in Oracle Rdb Release 7.2.1. All SQL dialects now accept FOR UPDATE for stronger row locking.

7.2.8 UPDATE ... WHERE CURRENT OF Assigns Incorrect Result Within IF Statement

Bug 2266270

In prior versions of Oracle Rdb, an UPDATE ... WHERE CURRENT OF executed within a conditional statement such as an IF THEN ELSE or CASE statement may assign the wrong result to the target column. This can only happen under the following conditions:

- There exists more than one UPDATE ... WHERE CURRENT OF statement, each in different branches of the IF or CASE statement.
- These UPDATE statements share a common expression.

The problem occurs because the evaluation of the common expression is performed within only one branch of the IF statement and is not visible for the other conditional branches. It is these other branches that will assign the incorrect result.

The following example shows the structure of such problem queries. The common expression in this example is A + B.

```
begin
for :x as each row of cursor y
    for select id, a, b from t_table
do
    if :x.id = 1 then
        update t_table set res = a + b + 1 where current of y;
    else
        update t_table set res = a + b where current of y;
    end if;
end for;
end;
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.2.9 Unexpected COSI−F−BUGCHECK Error Reported by SHOW Commands

Bug 5256548

In some cases, the SHOW INDEX (PARTITION), SHOW TABLE or SHOW STORAGE MAP commands may fail with a COSI−F−BUGCHECK error. This occurs when the storage area is in a UNIFORM format area and the name is exactly 31 octets in length. The error is due to the buffer being sized too small.

The following example shows the problem.

```
SQL> show table SAMPLE_TABLE;
Information for table SAMPLE_TABLE:
...
 Partition information for index:
  Partition: (1) SYS_P00150
   Storage Area: A_VERY_LONG_STORAGE_AREA_NAME_1
%COSI-F-BUGCHECK, internal consistency failure
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.2.10 ALTER DOMAIN Incorrectly Propagates DEFAULT Changes to Table Columns

Bug 5203032

In previous releases of Oracle Rdb, the ALTER DOMAIN ... DROP DEFAULT clause would propagate the change to all columns based on that domain. While this is correct in most cases, this action should not be performed when the column has its own DEFAULT which overrides that in the domain.

The side effect was that the DROP DEFAULT would cause the column's default value to be lost. However, commands such as SHOW TABLE (COLUMN) would still display the old value.

The following example shows this behavior. Here the domain includes a CHECK constraint to prevent NULL values being inserted. The DROP DEFAULT has left neither DEFAULT expression active and an attempt is made to insert NULL.

```
SQL> create domain D_TEST
cont>     integer
cont>     default -99
cont>     check (value is not null) not deferrable;
SQL>
SQL> create table T_TEST
cont>     (ident_column D_TEST default 1
cont>     ,subident_column D_TEST default 0
cont>     );
SQL>
SQL> insert into T_TEST default values;
1 row inserted
SQL> select * from T_TEST;
 IDENT_COLUMN   SUBIDENT_COLUMN
            1                0
1 row selected
SQL>
SQL> alter domain D_TEST
cont>     drop default;
SQL>
SQL> insert into T_TEST default values;
%RDB-E-NOT_VALID, validation on field IDENT_COLUMN caused operation to fail
SQL>
```

The correction is to redefine the column default using ALTER TABLE ... ALTER COLUMN ... DEFAULT.

This problem has been corrected in Oracle Rdb Release 7.2.1. ALTER DOMAIN will no longer propagate the DROP DEFAULT changes to any column that has an overriding DEFAULT clause at the column level.

7.2.11 Module Global Variables Limited to 64 DECLARE Statements

Bug 5346544

In previous versions of Oracle Rdb, a CREATE MODULE statement with more than 64 DECLARE statements for module global variables might bugcheck. The bugcheck summary would appear similar to the one shown below.

- %COSI-F-BUGCHECK, internal consistency failure
- Exception occurred at MEM_BUGCHECK + 00000010
- Called from RDMS$$CREATE_MODULE_INFO + 000012E4
- Called from RDMS$$CREATE_MODULE_INFO + 00000C50
- Called from RDMS$$RELEASE_DDL_VM_HNDLR + 0000130C

This problem was caused by a memory allocation error and has been corrected in Oracle Rdb Release 7.2.1. Oracle Rdb now supports a virtually unlimited number of module global variables.

7.2.12 Invalid Escape Sequence Not in SQLSTATE

Bug 5208821

The SQL standard requires that an invalid escape sequence be reported in SQLSTATE with a code of "22025". Additionally, Rdb is supposed to report this condition with a SQLCODE of −1040. In prior versions of Rdb, a SQLSTATE of "RR000" and a SQLCODE of −1 were reported instead.

For example, suppose a SQL$PRE/CC program contains the following query:

```
EXEC SQL SELECT COUNT(*) FROM CPBASE
         WHERE JUNK1 LIKE 'P%X%%X' ESCAPE 'X';
```

The escape sequence in the above query is invalid because it ends with an escape character. In prior versions, dynamic SQL would have reported the error with a very generic SQLCODE of −1 and SQLSTATE of "RR000". The correct SQLCODE of −1040 and SQLSTATE of "22025" are now reported.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.2.13 Incomplete Drop of Partitioned Indices with DROP STORAGE AREA ... CASCADE

Bug 5620530

In prior releases of Oracle Rdb, the ALTER DATABASE ... DROP STORAGE AREA ... CASCADE statement may not correctly update partitioned indices that refer to the rows deleted from a partitioned table.

The following example shows that some index nodes still reference the old rows. This is detected by RMU Verify.

```
$ DEFINE/USER RDMS$SET_FLAGS −
    "TEST_SYSTEM,STOMAP_STATS,INDEX_STAT,INTERNAL,ITEM_LIST"
$ SQL$
alter database
    filename SAMPLE_DATABASE
    drop storage area AREA_MAIN_07 cascade;
~As: Drop Storage Area "AREA_MAIN_07" Cascade
~As: ...area referenced by index: "INDEX1"
~As: ...area referenced by map: "MAP_TABLE_001"
~As: ...purge index "INDEX2"
~As: ...update the AIP for larea=65 (table)
~As: ...update the AIP for larea=77 (index)
~H Extension (VERIFY CONSTRAINTS) Item List:
0000 (00000) RDB$K_EXT_VFYC_EXCLUDE_UNIQUE
0003 (00003) RDB$K_EXT_VFYC_TABLE_NAME "TABLE_001"
000F (00015) RDB$K_INFO_END
$
$ RMU/VERIFY/INDEX SAMPLE_DATABASE
%RMU−W−CANTFINDLAREA, cannot locate logical area 65 in area inventory page list
%RMU−E−BDLAREADY, error readying logical area with dbid 65
%RMU−I−READYDATA, ready needed for data record at 65:5:0
%RMU−I−BTRNODDBK, Dbkey of B−tree node is 89:3:0
```

```
%RMU-W-BTRVFYPRU, B-tree verification pruned at this dbkey
%RMU-I-BTRPARROO, root dbkey of b-tree partition in AREA_INDEX_07 is 89:3:0
$
```

This problem has been corrected in Oracle Rdb Release 7.2.1. The only workaround for this problem is to drop and recreate the affected indices.

7.2.14 Unexpected LENMISMAT Warnings when Using TRANSLATE ... USING Function

Bug 5629307

In prior versions Oracle Rdb, the result length of the TRANSLATE (... USING ...) function was overestimated by SQL. In some cases, this caused unexpected and erroneous warnings to be issued.

The following example shows this on a simple column. There should be no warnings from this command.

```
SQL> set character length 'characters';
SQL> create table utest (u5 char(10) character set unicode) ;
SQL> show table (column) utest
Information for table UTEST

Columns for table UTEST:
Column Name                    Data Type        Domain
-----------                    ---------        ------
U5                             CHAR(10)
         UNICODE 10 Characters,  20 Octets

SQL> insert into utest values ( translate ('A' using rdb$unicode ) ) ;
1 row inserted
SQL> insert into utest values ( translate ('AB' using rdb$unicode ) ) ;
1 row inserted
SQL> insert into utest values ( translate ('ABC' using rdb$unicode ) ) ;
1 row inserted
SQL> insert into utest values ( translate ('ABCD' using rdb$unicode ) ) ;
1 row inserted
SQL> insert into utest values ( translate ('ABCDE' using rdb$unicode ) ) ;
1 row inserted
SQL> insert into utest values ( translate ('ABCDEF' using rdb$unicode ) );
%SQL-W-LENMISMAT, Truncating right hand side string for assignment to column U5
1 row inserted
SQL> insert into utest values ( translate ('ABCDEFG' using rdb$unicode ) );
%SQL-W-LENMISMAT, Truncating right hand side string for assignment to column U5
1 row inserted
SQL> commit;
```

This problem has been corrected in Oracle Rdb Release 7.2.1. SQL now uses the octet length of the maximum size character in the character set for the estimation. While it is now less likely that TRANSLATE will issue unnecessary LENMISMAT warnings, SQL may not know the final translation of the source character string, and for some variable length character sets the warning may be justified even when the assignment succeeds without truncation.

7.2.15 Unexpected Error from UNION Containing NULL Expression

Bug 5645199

In prior versions of Oracle Rdb, a UNION operator that specified NULL in the select list of the first leg might derive an invalid data type for the common data type. This could result in garbled results (as shown in the example below) or produce an error at run time.

```
%RDB-E-ARITH_EXCEPT, truncation of a numeric value at runtime
-SQL-F-UNSDTPCVT, Unsupported data type conversion
```

The following example shows the incorrect results.

```
SQL> select null from ntab
cont> union
cont> select d1 from dtab;
 D1
 @L.oGe%.....(x/z(x/z...
 NULL
2 rows selected
```

Workarounds for this problem include: wrapping a CAST expression around NULL and specifying a data type that is compatible with the other legs of the UNION, or reversing the select expressions so that the NULL expression is processed last. The next example shows the expected result.

```
SQL> select d1 from dtab
cont> union
cont> select null from ntab;
 D1
  6-NOV-2006 16:16:52.62
 NULL
2 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.2.16 Inconsistent Data Type Assignment to IS NULL Expression

Bug 5484239

In prior releases of Oracle Rdb, Dynamic SQL would assign a data type to a parameter marker in an IS NULL clause based on a nearby expression. In many cases, this data type was not consistently applied.

The following example shows a Dynamic SQL application prompting for data from the user. In some cases the ? IS NULL requests an integer (input 4 and 8) and at other times it requests a char(30) input.

```
Enter statement:
create table CUSTOMERS
    (id INTEGER
    ,ORDERID INTEGER
    ,NAME CHAR(30)
    );
```

```
Enter statement:
update CUSTOMERS
set ID=?, ORDERID=?, NAME=?
where (ID= ? OR (ID IS NULL AND ? IS NULL))
  and (ORDERID= ? OR (ORDERID IS NULL AND ? IS NULL))
  and (NAME= ? OR (NAME IS NULL AND ? IS NULL));
[9 fields]
 0/ID/Integer: 12345
 1/ORDERID/Integer: 345
 2/NAME/Char(30/30): Jones
 3/ID/Integer: 12355
 4//Integer: 0
 5/ORDERID/Integer: 344
 6//Char(30/30):
 7/NAME/Char(30/30): Lee
 8//Integer: 0
Enter statement:
```

This problem has been corrected in Oracle Rdb Release 7.2.1. In this release, SQL will assign the default type (that is VARCHAR(2000)) as the data type for the expression "? IS NULL".

7.2.17 Table Synonym Not Used by Query Outlines

In prior releases of Oracle Rdb, a synonym created for a table using CREATE SYNONYM or RENAME TABLE was not recognized by the query outline at runtime. A message similar to the following would be reported when the 'STRATEGY' flag was specified with the SET FLAGS statement.

```
SQL> select a from t000 order by a;
~S: Outline "QO_A" used
~S: Outline/query mismatch; assuming T000 0 renamed to TABLE_000 0
Tables:
  0 = TABLE_000
Index only retrieval of relation 0:TABLE_000
  Index name  T000_INDEX [0:0]
0 rows selected
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.1. Oracle Rdb now compares the result target table instead of just comparing the name referenced in the query outline with that of the table in the query.

7.2.18 Unexpected UNSDTPCVT Error During String Concatenation

Bug 5584169

When a zero length character string is concatenated with another string, SQL unexpectedly reports a UNSDTPCVT error. This is due to an attempt to apply Oracle semantics to the query. This error only occurs when the dialect is set to either ORACLE LEVEL1 or ORACLE LEVEL2.

The following example shows a failing query due to this problem.

```
SQL> set dialect 'oracle level1';
SQL> select 'test'||'' from rdb$database;
%SQL-F-UNSDTPCVT, Unsupported data type conversion
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.2.19 Parameter for LIKE Pattern Sized Too Small

Bug 4179408

If Dynamic SQL is processing a query that uses a parameter marker for the LIKE pattern, then it currently assumes the parameter is the same length as the source string. However, a pattern such as '%123%' is perfectly valid for use in matching against a CHAR(4) column (matching leading 123 or trailing 123) but the assumed data type causes the pattern to be truncated by Oracle Rdb and consequently not all values will be matched.

The following example shows the old behavior. All three rows should be matched by the query.

```
$ run test$tools:tester
Enter statement:
attach 'filename sql$database';
Enter statement:
create table LL (val char(4));
Enter statement:
insert into LL (val) values ('1234');
Enter statement:
insert into LL (val) values ('1235');
Enter statement:
insert into LL (val) values ('0123');
Enter statement:
select * from LL where val like ?;
[1 fields]
 0/VAL/Varchar(4/8): %123%
Enter statement:
```

The following example shows the corrected behavior and increased length of the parameter marker data type.

```
$ run test$tools:tester
Enter statement:
attach 'filename sql$database';
Enter statement:
create table LL (val char(4));
Enter statement:
insert into LL (val) values ('1234');
Enter statement:
insert into LL (val) values ('1235');
Enter statement:
insert into LL (val) values ('0123');
Enter statement:
select * from LL where val like ?;
[1 fields]
 0/VAL/Varchar(8/12): %123%
 0/VAL: 1234
 0/VAL: 1235
 0/VAL: 0123
Enter statement:
```

This problem has been corrected in Oracle Rdb Release 7.2.1. SQL now assumes that the like pattern is twice the size of the source string, or if the ESCAPE clause is present, it assumes three times the size. This should allow room for most pattern strings. If this sizing is still too small, use CAST(? AS VARCHAR(n)) to size the parameter to a more precise length.

7.3 RMU Errors Fixed

7.3.1 RMU/BACKUP/AFTER Ignores Default Filename When /EDIT_FILENAME Included

Bug 5464971

When an RMU/BACKUP/AFTER command was issued and no output filename was given and the /EDIT_FILENAME qualifier was included, the default journal filename would not be used when creating the backup file. For example:

```
$ RMU/BACKUP/AFTER/LOG -
  /EDIT_STRING=("_", VNO, "_", YEAR,MONTH,DAY_OF_MONTH) -
  MF_PERSONNEL .AIJ
.
.
.
%RMU-I-LOGCREBCK, created backup file DEV:[DIR]_0_20060829.AIJ;1
```

In the above example, the journal filename was "J1" and that name should have been used as the prefix for the backup filename but instead only the contents of the edit string were used to construct the filename.

This problem can be avoided by explicitly providing the backup output filename in the backup command.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.3.2 Possible RMU COLLECT OPTIMIZER Workload Statistics Memory Corruption

Bug 5436532

A system access violation could occur when using the RMU/COLLECT OPTIMIZER command to collect WORKLOAD statistics for an Rdb database. This problem was caused by a memory corruption problem that could happen when bits were set outside of a bitmap table used to calculate workload statistics for tables with a larger number of rows. This has been fixed and bits can no longer be set outside the bounds of the bitmap table. This problem only occurs for Oracle Rdb RMU V7.2.

The following example shows the system access violation which could occur when workload statistics were collected.

```
$ RMU/COLLECT OPTIMIZER/STATISTICS=(WORKLOAD) test_datatbase
%SYSTEM-F-ACCVIO, access violation, reason mask=04,
virtual address=000000000000000C, PC=0000000000501D2C, PS=0000001B
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
%RMU-I-BUGCHKDMP, generating bugcheck dump file
device:[directory]RMUBUGCHK.DMP
```

The partial workaround for this problem is to first use the /TABLE qualifier on the RMU/COLLECT OPTIMIZER command to see which tables cause this problem and then to use the /EXCLUDE_TABLES

qualifier to not collect workload statistics for those tables.

```
$ RMU/COLLECT OPTIMIZER/STATISTICS=(WORKLOAD)-
 /EXCLUDE_TABLE=problem_table test_database
$
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.3.3 RMU VERIFY Memory Corruption

Bugs 5349580 and 5276155

Because of database corruption, it was possible for a counter used to allocate memory for the structures which guide the database verify to be too small. This can happen if the corruption causes queries of the system tables to return inconsistent data. One cause of this could be corruption of Area Bitmap Pages for uniform areas.

When other queries were made to the database to fill entries in the verify structures for tables, indexes, segmented strings, etc., entries put in the verify structures for the various database objects could go beyond the memory allocated for the structures. This caused memory corruption which could result in a system error and an RMU/VERIFY bugcheck.

The database corruption causing inconsistent data to be returned from the system tables cannot be repaired by the verify. But now checks are made to prevent an overrun of the memory size allocated for the verify structures and a warning message will be output that not all of the database objects will be verified because of a problem accessing the system tables but that the verify will continue. Since this is a case where queries to the database cannot be trusted and there is no way to tell what is causing this inconsistency at the point where it happens at the start of the verify, this gives a chance for the verify to continue so it can show the problem.

The following example shows a case where an RMU/VERIFY of a database encountered corruption which caused a memory overrun when loading the verify structures. This caused memory corruption which resulted in repeated system reserved operand faults. The verify tried to continue but when it could not load the structures needed to go on with the verify, the verify aborted and output a bugcheck dump because of an unexpected system error.

```
$ RMU/VERIFY/ALL device:[directory]database.rdb
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%SYSTEM-W-ROPRAND, reserved operand fault at PC=000000000032E178,
PS=0000001B
%RMU-E-ERRRDBIND, error accessing RDB$INDICES relation
%RMU-I-PARTLVFY,  continuing partial verification
%RMU-F-ABORTVER,  fatal error encountered; aborting verification
%SYSTEM-F-ROPRAND, reserved operand fault at PC=000000000032E178,
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_VER, Fatal error for VERIFY operation at 21-JUN-2006
08:03:57.63
```

The following example shows the corrected behavior. The memory overrun is prevented and a warning message is output for the particular database object type stating that there was a problem loading some of the information to completely verify all objects of that type. The verify continues so that any diagnostics it returns

can help identify the problem. In the case of the Area Bit Map page corruption, an RMU/REPAIR can be executed to fix the corruption problem.

```
$ RMU/VERIFY/ALL device:[directory]database.rdb
%RMU-W-NOTALLDAT, Not all data for database TABLES can be loaded from
system tables - verify continuing.
%RMU-W-ABMBITERR, inconsistency between spam page 9475371 and bit 918
in area bitmap in larea 1 page 6
%RMU-W-ABMBITERR, inconsistency between spam page 9490631 and bit 932
in area bitmap in larea 1 page 6
%RMU-W-ABMBITERR, inconsistency between spam page 9491721 and bit 933
in area bitmap in larea 1 page 6
%RMU-E-BADABMPAG,       error verifying ABM pages
$ RMU/REPAIR/ABM/AREA=UNIFORM_AREA device:[directory]DATABASE.RDB
%RMU-I-FULBACREQ, A full backup of this database should be performed
 after RMU REPAIR
$ RMU/VERIFY/ALL device:[directory]DATABASE.RDB
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.3.4 Unexpected DLMNOTFND When Leading Characters of Suffix Appear in Data

Bug 4245634

In prior versions of Oracle Rdb, RMU Load would report an error if the data included the leading character from the SUFFIX. This occurred with FORMAT=DELIMITED when a SUFFIX option was specified. For example, the name "SMITH, ANDREW", in the sample data below, is followed by a "/" which is also the leading character of the SUFFIX defined on the RMU Load command line.

```
/:/key3/:/,/:/SMITH, ANDREW//:/,/:/Z/:/&END&
```

This causes the RMU Load to fail as shown below.

```
$ RMU/LOAD/RECORD_DEFINITION=(-
    FILE=NAMES_TABLE.RRD,-
    FORMAT=DELIMITED_TEXT, -
    PREFIX="/:/",-
    SUFFIX="/:/",-
    TERMINATOR="&END&",-
    NULL="")-
    LOAD_TEST NAMES_TABLE NAMES.DAT
%RMU-F-DLMNOTFND, Separator (,) not found for column 2 of row 1 in the input.
%RMU-I-DATRECREAD,  3 data records read from input file.
%RMU-I-DATRECSTO,   0 data records stored.
%RMU-I-DATRECREJ,   0 data records written to exception file.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 14-JUL-2006 12:49:18.79
```

This problem has been corrected in Oracle Rdb Release 7.2.1. RMU Load now correctly scans ahead for the correct SUFFIX.

7.3.5 Unexpected Failure of RMU Load When the NULL String Appears With Column Data

Bug 4865227

In prior releases of Oracle Rdb, the RMU Load command would fail if it detected the NULL string within column data. This occurs with FORM=DELIMITED and specifying the NULL option to the RECORD_DEFINITION qualifier. The following example shows a simple case.

```
$ RMU/UNLOAD/RECORD=(FILE=T1.RRD,FORMAT=DELIMITED,-
        SEPARATOR="|",PREFIX="",SUFFIX="",NULL="***") -
        SAMPLE_DB T1 T1.DAT
%RMU-I-DATRECUNL,   1 data records unloaded.
$ TYPE T1.DAT
abcde|N***N     |z
$ RMU/LOAD/RECORD=(FILE=T1.RRD,FORMAT=DELIMITED,-
        SEPARATOR="|",PREFIX="",SUFFIX="",NULL="***") -
        SAMPLE_DB T1 T1.DAT
  DEFINE FIELD C1 DATATYPE IS TEXT SIZE IS 5.
  DEFINE FIELD C2 DATATYPE IS TEXT SIZE IS 10.
  DEFINE FIELD C3 DATATYPE IS TEXT SIZE IS 1.
  DEFINE RECORD T1.
     C1 .
     C2 .
     C3 .
  END T1 RECORD.
%RMU-F-UNEXPDELIM, Unexpected delimiter encountered (***) in row 1 of input
%RMU-I-DATRECREAD,  1 data records read from input file.
%RMU-I-DATRECSTO,   0 data records stored.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 14-JUL-2006 00:57:40.42
```

RMU Load should not have reported an error in this case as the data is not ambiguous.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.3.6 Unexpected INVALID_BLR Error Reported For RMU Load

Bug 4040175

In prior releases of Oracle Rdb, the RMU Load command would accept a record definition file that contained the data types PACKED DECIMAL, UNSIGNED NUMERIC and LEFT SIGNED NUMERIC. This would lead to errors similar to this example:

```
$ rmu/load abc temp_table /rec=(format=text,file=z.rrd) z.dat
%RDB-E-INVALID_BLR, request BLR is incorrect at offset 8
%RMU-I-DATRECREAD,  0 data records read from input file.
%RMU-I-DATRECSTO,   0 data records stored.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 13-JUL-2006 20:50:23.81
```

These types are not supported by the Oracle Rdb server and should not be allowed by RMU Load. This release will correctly diagnose the use of these types.

```
%RMU-F-UNSSUPDAT, Unsupported data type: 16
%RMU-I-DATRECSTO,   0 data records stored.
%RMU-F-FTL_LOAD, Fatal error for LOAD operation at 13-JUL-2006 20:50:40.17
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.3.7 RMU /COPY, /MOVE and /RESTORE Could Corrupt ABM Pages

Bug 5276155

For RMU COPY, MOVE and RESTORE, the Area Bit Map pages for uniform storage areas could be incorrectly set if there was a chain of multiple ABM bitmap pages for a logical area and the previous ABM page of the chain had NO bits set but subsequent ABM pages in the chain had bits set. This could happen if there were enough SPAM pages in the uniform storage area so that not all SPAM PAGES could be represented by the bit map in a single ABM page so multiple ABM pages, each with a bitmap, were required to represent the SPAM pages for each logical area. If one of the ABM pages (but not the last) for a logical area had NO bits set, then any ABM pages following it that had bits set before the RMU MOVE, COPY or RESTORE would have all bits cleared after the RMU COPY, MOVE or RESTORE. If a bit is not set for an ABM page for a logical area, then Rdb will not retrieve any database data pages controlled by that SPAM page. Note that this can only happen for the case described and only for uniform storage areas. This will be detected by RMU/VERIFY and can be fixed by RMU/REPAIR/ABM/AREAS. This problem has been fixed and now the bits in the ABM bitmaps will be correctly set in this case.

The following example shows a case where an RMU/COPY of a database causes this problem. The problem is then detected by an RMU/VERIFY and fixed by an RMU/REPAIR.

```
$ RMU/COPY/NOLOG/DIRECTORY=device:[directory] DATABASE.RDB
$ RMU/VERIFY/AREAS/LAREAS device:[directory]DATABASE.RDB
%RMU-W-ABMBITERR, inconsistency between spam page 8650241 and bit 161 in
area bitmap in larea 113 page 8650819
%RMU-W-ABMBITERR, inconsistency between spam page 8651331 and bit 162 in
area bitmap in larea 113 page 8650819
%RMU-W-ABMBITERR, inconsistency between spam page 8652421 and bit 163 in
area bitmap in larea 113 page 8650819
%RMU-W-ABMBITERR, inconsistency between spam page 8653511 and bit 164 in
area bitmap in larea 113 page 8650819
%RMU-W-ABMBITERR, inconsistency between spam page 8654601 and bit 165 in
area bitmap in larea 113 page 8650819
%RMU-W-ABMBITERR, inconsistency between spam page 8655691 and bit 166 in
area bitmap in larea 113 page 8650819
%RMU-W-ABMBITERR, inconsistency between spam page 8656781 and bit 167 in
area bitmap in larea 113 page 8650819
%RMU-E-BADABMPAG,       error verifying ABM pages
$ RMU/REPAIR/ABM/AREA=UNIFORM_AREA device:[directory]DATABASE.RDB
%RMU-I-FULBACREQ, A full backup of this database should be performed
 after RMU REPAIR
$ RMU/VERIFY/AREAS/LAREAS device:[directory]DATABASE.RDB
$
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.3.8 RMU Unload May Truncate REAL Data When Delimited or Text Format Used

Bug 4297346

In prior releases of Oracle Rdb, an RMU Unload of a virtual column might in some rare cases result in the data being truncated.

This could occur if the computation was a table COMPUTED BY or view column that calculated the result using LEAST, GREATEST, CASE, NULLIF, COALESCE, NVL, NVL2, or DECODE expression. For this problem to occur, the computation must include one expression that resulted in REAL (F Floating) and another that resulted in SMALLINT. Unfortunately, Oracle Rdb was promoting the result to DOUBLE PRECISION (G Floating) prior to converting the value to a string value. When the value was delivered to RMU Unload, the target string, which was sized correctly for a REAL string value, was too small for the resulting DOUBLE PRECISION string and the result was truncated.

The workaround for this problem is to include an explicit CAST(... AS REAL) in the COMPUTED BY or view column definition.

The following output shows the truncation of the column:

```
003537||| 9.7500000E+01|||975||| 9.750000000000
```

The expected result would include the exponent.

```
003537||| 9.7500000E+01|||975||| 9.750000000000000E+001
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.3.9 Incomplete Multischema Database Support in RMU Extract

Bugs 5558122, 5568079, and 5568040

In prior releases of Oracle Rdb, RMU Extract had incomplete support for multischema databases. With this release, the following corrections have been made to RMU Extract.

- Extracted query outlines did not output the names of tables, indices or modules correctly. Only the STORED NAME was used which caused the generated script to fail.
- Extracted views included the STORED NAME IS clause but the name may not have been delimited when it contained lowercase characters or different character set values. This also caused errors when executing the generated script.
- Most other objects did not include the STORED NAME IS clause at all and so there was possibly conflict with names generated by SQL for tables and any view definitions that were subsequently extracted.

These problems have been corrected in Oracle Rdb Release 7.2.1.

7.3.10 RMU Extract Item=Protections Did Not Consistently Extract Protections

Bug 5225643

In previous versions of Oracle Rdb, the RMU Extract Item=Protections output for a table might include the unused OPERATOR privilege for the table or omit the REFERENCES privilege for a module, function or procedure. These errors are harmless but the second error could prevent any routine created by the generate script being a target for synonym.

This problem has been corrected in Oracle Rdb Release 7.2.1. RMU now consistently outputs the protections for all objects.

7.3.11 RMU/CONVERT Bugchecks in PIO$LOCK_PAGE When Statistics Disabled

If statistics were disabled while executing an RMU/CONVERT command, the RMU utility would bugcheck with a stack footprint similar to the following:

```
***** Exception at 0000000000719708 : RMU72\PIO$DEMOTE_PAGE + 000001A8
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual
address=0000000000000000, PC=0000000000719708, PS=0000001B
Saved PC = 0000000000722E14 : RMU72\PIOUTL$EMPTY_ONE_BUFFER + 000002B4
Saved PC = 000000000071D654 : RMU72\PIOFETCH$WITHIN_DB_HNDLR + 00000134
Saved PC = FFFFFFFF81104EC8 : Image LIBOTS + 00008EC8
Saved PC = FFFFFFFF800A693C : symbol not found
***** Exception at 000000000071C020 : RMU72\PIO$LOCK_PAGE + 00000320
Saved PC = 000000000071E114 : RMU72\PIOFETCH$WITHIN_DB + 00000924
Saved PC = 000000000071B444 : RMU72\PIOFETCH$FETCH + 000002E4
Saved PC = 000000000071A4E4 : RMU72\PIO$FETCH + 000008F4
```

The same problem might also occur when an implied conversion was done by restoring a backup that was made with a prior version of Oracle Rdb.

This problem can be demonstrated with the following commands:

```
$ @SYS$LIBRARY:RDB$SETVER 71
Current PROCESS Oracle Rdb environment is version V7.1-441 (MULTIVERSION)
Current PROCESS SQL environment is version V7.1-441 (MULTIVERSION)
Current PROCESS Rdb/Dispatch environment is version V7.1-441 (MULTIVERSION)
$ MCR SQL$71
SQL> CREATE DATABASE FILENAME TEST;
SQL> EXIT
$
$ DEFINE RDM$BIND_STATS_ENABLED 0
$
$ @SYS$LIBRARY:RDB$SETVER 72
Current PROCESS Oracle Rdb environment is version V7.2-010 (MULTIVERSION)
Current PROCESS SQL environment is version V7.2-010 (MULTIVERSION)
Current PROCESS Rdb/Dispatch environment is version V7.2-010 (MULTIVERSION)
$ RMU/CONVERT/NOCONFIRM TEST
%RMU-I-RMUTXT_000, Executing RMU for Oracle Rdb V7.2-010
%RMU-I-LOGCONVRT, database root converted to current structure level
%RMU-I-BUGCHKDMP, generating bugcheck dump file dev:[dir]RMUBUGCHK.DMP;
```

This problem can be avoided by deassigning the RDM$BIND_STATS_ENABLED logical prior to executing the RMU/CONVERT command.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.3.12 RMU/RESTORE/AREA/ONLINE Was Unnecessarily Locking RDB$SYSTEM for PROTECTED READ

Bug 5203722

RMU/RESTORE/AREA/ONLINE was locking the RDB$SYSTEM area in Protected Read mode even though RDB$SYSTEM was not one of the storage areas being restored. RMU/RESTORE/AREA/ONLINE does have to access the system area in order to read and possibly change the Area Inventory Pages if they are inconsistent. The Area Inventory Pages are located in the system area. However, RMU was unnecessarily locking the system area even when only mixed format storage areas, which do not have AIP pages referencing them, were being restored.

Now the locking and accessing of the system area AIP pages has been changed for an online by area restore. If all areas being restored are of MIXED format, the system area is not locked if it is not one of the areas being restored. If any uniform areas are being restored online, then the system area will be readied in Concurrent Write mode instead of Protected Read mode to provide more concurrency during the restore. For uniform areas, the AIP pages need to be accessed to help reconstruct the Area Bit Map and Space Management Pages and possibly changed if they are inconsistent.

Note that any of the areas actually being restored which are named in the restore command, including the system area, will continue to be locked for Exclusive Update.

The following example shows the online by area restore commands affected by these changes. If the restore references only mixed areas, there is no locking of the system area. If the restore references any uniform areas, there will now be Concurrent Write locking of the system area where there was formerly Protected Read locking of the system area. If the system area is itself being restored, it will continue to be locked for exclusive update.

```
$ RMU/RESTORE/ONLINE/AREA/NOLOG/DIR=DEVICE:[DIRECTORY] -
         DEVICE:[DIRECTORY]DATABASE.RBF MIXED_1_AREA, MIXED_2_AREA
$ RMU/RESTORE/ONLINE/AREA/NOLOG/DIR=DEVICE:[DIRECTORY] -
         DEVICE:[DIRECTORY]DATABASE.RBF MIXED_1_AREA, MIXED_2_AREA, -
                        UNIF_1_AREA, UNIF_2_AREA
$ RMU/RESTORE/ONLINE/AREA/NOLOG/DIR=DEVICE:[DIRECTORY] -
         DEVICE:[DIRECTORY]DATABASE.RBF MIXED_1_AREA, MIXED_2_AREA,-
                        UNIF_1_AREA, UNIF_2_AREA, RDB$SYSTEM
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.3.13 Failure of RMU /BACKUP of Database With Very Large Storage Area Count and Small Specified /BLOCK_SIZE Value

Bug 5376038

Previously, databases with a very large number of storage areas and a specified small value for /BLOCK_SIZE might cause RMU /BACKUP to fail. An internal buffer could overflow the output file block size and would either write a record that could not be read during recovery or could corrupt memory and cause an ACCVIO bugcheck during the backup. For some versions, the buffer overflow would be detected at run time and the RMU /BACKUP operation would exit with a bugcheck dump with a "footprint" similar to the following:

```
$ RMU/BACKUP FOO.RDB FOO.RBF /BLOCK_SIZE=2048
***** Exception at 003E95F0 : RMUBCK$BACKUP_SUMMARY + 00000270
%COSI-F-BUGCHECK, internal consistency failure
Saved PC = 003E8584 : RMUBCK$BF_BACKUP_THREAD + 00000494
Saved PC = 008302FC : RMUIO$TERMINATE_THREAD + 00000040
```

```
Saved PC = 008304B8 : RMUIO$FIREWALL + 00000040
Saved PC = 00830478 : RMUIO$FIREWALL + 00000000
Saved PC = 003A18E4 : RMU_DISPATCH + 00000434
Saved PC = 003A1008 : RMU_STARTUP + 000004E8
Saved PC = 001E002C : RMU$MAIN + 0000002C
```

This problem has been corrected in Oracle Rdb Release 7.2.1. RMU /BACKUP now correctly adjusts the backup block size as needed to accommodate the number of database storage areas and page sizes.

7.3.14 RMU Extract Reports BAD_CODE Error for BITSTRING Function

In prior releases of Oracle Rdb, RMU Extract would generate a BAD_CODE error when trying to extract a BITSTRING function nested within a CASE, ABS, COALESCE, DECODE, NULLIF, NVL, or NVL2 function. The following example shows the reported error.

```
%RMU-F-BLRINV, internal error - BLR string 83 for . is invalid
-RDMS-E-BAD_CODE, corruption in the query string
%RMU-F-FTL_RMU, Fatal error for RMU operation at 10-JUL-2006 03:27:48.68
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.3.15 Incorrect SQL Syntax Generated for Views Containing UNION and GROUP BY Clauses

Bugs 5666305 and 5672460

In prior releases, RMU Extract would incorrectly extract view definitions by including an asterisk "*" following the select value list when a UNION included a branch containing a GROUP BY clause.

This example shows the original view definition.

```
SQL> create view SAMPLE_VIEW (a, b, c)
cont>     as
cont>     select last_name, first_name, middle_initial
cont>     from candidates
cont>     group by last_name, first_name, middle_initial
cont> union
cont>     select last_name, first_name, middle_initial
cont>     from employees
cont>     group by last_name, first_name, middle_initial
cont> ;
```

This is the output from RMU Extract showing the incorrect syntax.

```
   .
   .
   .
create view SAMPLE_VIEW
    (A,
     B,
     C) as
    select C3.LAST_NAME, C3.FIRST_NAME, C3.MIDDLE_INITIAL
        * from CANDIDATES C3
```

```
        group by C3.LAST_NAME, C3.FIRST_NAME, C3.MIDDLE_INITIAL
    union
    select C5.LAST_NAME, C5.FIRST_NAME, C5.MIDDLE_INITIAL
        * from EMPLOYEES C5
        group by C5.LAST_NAME, C5.FIRST_NAME, C5.MIDDLE_INITIAL;
.
.
.
```

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.4 LogMiner Errors Fixed

7.4.1 RMU /UNLOAD /AFTER_JOURNAL Incorrect NULL Bit Setting

Bug 5256671

In prior Oracle Rdb releases, certain modifications and structures of database table definitions could result in the LogMiner improperly processing the null column information.

The following example demonstrates one possible cause of incorrect NULL processing. Note that in the extracted records, incorrect columns are indicated as NULL.

```
$ DEFINE /NOLOG SQL$DATABASE FOO
$ SQL$
    CREATE DATA FILE SQL$DATABASE
        NUMBER OF BUFFERS 100 NUMBER OF CLUSTER NODES 1;
    DISCONNECT ALL;
    ALTER DATA FILE SQL$DATABASE
        JOURNAL ENA (FAST COMMIT ENA) ADD JOURNAL J1 FILE J1;
    CREATE TABLE T1 (
        I1 INT,I2 INT,I3 INT,I4 INT,I5 INT,I6 INT,
        I7 INT,I8 INT,I9 INT,I10 INT,I11 INT,I12 INT);
    CREATE STORAGE MAP M1 FOR T1 DISABLE COMPRESSION;
    COMMIT;
    ALTER TABLE T1 DROP COLUMN I5;
    COMMIT;
    ALTER TABLE T1
        ADD COLUMN I13 INT
        ADD COLUMN I14 INT
        ADD COLUMN I15 INT
        ADD COLUMN I16 INT
        ADD COLUMN I17 INT;
    COMMIT;
    ALTER TABLE T1 DROP COLUMN I17;
    ALTER TABLE T1 ADD COLUMN I18 INT;
    COMMIT;
    ALTER TABLE T1 DROP COLUMN I18;
    COMMIT;

    DISCONNECT ALL;
    EXIT;
$ RMU/SET LOGMINER/ENABLE/NOLOG SQL$DATABASE
$ RMU/BACKUP/AFTER/NOLOG SQL$DATABASE NLA0:BAR
$ RMU/BACKUP/NOLOG/NOCRC/NOCHECKSUM SQL$DATABASE NLA0:BAR
$ SQL$
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,9,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (NULL,2,3,4,6,7,8,9,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,NULL,3,4,6,7,8,9,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,NULL,4,6,7,8,9,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,NULL,6,7,8,9,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,NULL,7,8,9,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,NULL,8,9,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,7,NULL,9,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,NULL,10,11,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,9,NULL,11,12,13,14,15,16);
```

```
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,9,10,NULL,12,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,9,10,11,NULL,13,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,9,10,11,12,NULL,14,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,9,10,11,12,13,NULL,15,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,9,10,11,12,13,14,NULL,16);
    INSERT INTO T1 VALUES (1,2,3,4,6,7,8,9,10,11,12,13,14,15,NULL);
    COMMIT;
$ RMU/BACKUP/AFTER/NOLOG SQL$DATABASE B1
$ RMU/UNLOAD/AFTER/NOLOG/FORMAT=DUMP SQL$DATABASE B1 -
    /TABLE=(NAME=T1,OUTPUT=T1.DAT)
$ SEARCH T1.DAT NULL
I13                             : NULL
I9                              : NULL
I13                             : NULL
I10                             : NULL
I13                             : NULL
I11                             : NULL
I13                             : NULL
I12                             : NULL
I13                             : NULL
I13                             : NULL
I14                             : NULL
I13                             : NULL
I15                             : NULL
I13                             : NULL
I16                             : NULL
I13                             : NULL
I13                             : NULL
I13                             : NULL
I13                             : NULL
I13                             : NULL
I13                             : NULL
I13                             : NULL
I13                             : NULL
$ EXIT
```

This problem has been corrected in Oracle Rdb Release 7.2.1. The LogMiner now correctly tracks the length of internal null information structures based on the table definition.

7.4.2 RMU /UNLOAD /AFTER_JOURNAL AERCP_LEN Field Incorrect In Text Format

Bug 5617814

Previously, when using the TEXT or DELIMITED_TEXT output format, the AERCP_LEN field of the commit record content would be incorrect.

For example, when using delimited text format, the following information might be returned in a commit record:

```
"C",
"",
    .
    .
    .
"16",
"7169",
"00000000-0000-0000-0000-000000000000",
```

```
"000000000000038F000000000000038F0000016000000000000001C01"
```

Here, the field value "16" is the RM_TID_LEN and the "7169" value is the incorrect AERCP_LEN field.

This problem has been corrected in Oracle Rdb Release 7.2.1. The correct value for the AERCP_LEN field is now returned. Note that the AERCP_LEN value is 28 and this represents the unformatted binary length of the AERCP structure and not the length of the text formatted field.

7.5 Row Cache Errors Fixed

7.5.1 RMU/RECOVER of Journaled Row Cache Changes Corrupts Database

Bug 5469750

If a database had row cache parameters changed, and the database was restored and recovered, the resulting database would be corrupt. Sometimes the RMU/RECOVER process would fail as well, and occasionally the Oracle Rdb monitor process would fail.

The following script demonstrates the problem.

```
$
$ ! Create a database with a few basic caches.
$
$ SQL$
  CREATE DATABASE FILENAME TEST
  NUMBER OF CLUSTER NODES 1
  RESERVE 4 STORAGE AREAS
  RESERVE 3 JOURNALS
  RESERVE 3 CACHE SLOTS
  ROW CACHE IS ENABLED
  CREATE STORAGE AREA RDB$SYSTEM FILENAME TEST
  CREATE STORAGE AREA TEST_A1 FILENAME TEST_A1
  CREATE STORAGE AREA TEST_A2 FILENAME TEST_A2
  CREATE STORAGE AREA TEST_A3 FILENAME TEST_A3
  CREATE CACHE TEST_A1 CACHE SIZE 100 ROWS ROW LENGTH 100 BYTES
  CREATE CACHE TEST_A2 CACHE SIZE 200 ROWS ROW LENGTH 200 BYTES
  CREATE CACHE TEST_A3 CACHE SIZE 300 ROWS ROW LENGTH 300 BYTES;
  DISCONNECT ALL;

  ALTER DATABASE FILENAME TEST
    ADD JOURNAL TEST_J1 FILENAME SYS$DISK:[]TEST_J1.AIJ
    ADD JOURNAL TEST_J2 FILENAME SYS$DISK:[]TEST_J2.AIJ
    ADD JOURNAL TEST_J3 FILENAME SYS$DISK:[]TEST_J3.AIJ
    JOURNAL IS ENABLED (FAST COMMIT ENABLED);
%RDMS-W-DOFULLBCK, full database backup should be done to ensure future recovery
  EXIT;
$
$ ! Save away the original cache configuration.
$
$ RMU/BACKUP/NOLOG TEST TEST
$ RMU/DUMP/HEADER=BRIEF/OUTPUT=BEFORE.TXT TEST
$
$ ! Alter a cache parameter.
$
$ SQL$
  ALTER DATABASE FILENAME TEST ALTER CACHE TEST_A2 ROW LENGTH 400 BYTES;
  DISCONNECT ALL;

-- Delete the database

  DROP DATABASE FILENAME TEST;
  EXIT;
$
```

```
$ ! Restore the database.  RMU will automatically recover the journals.
$
$ RMU/RESTORE/NOCDD/NOLOG TEST
%RMU-I-AIJRSTAVL, 3 after-image journals available for use
%RMU-I-AIJRSTMOD, 1 after-image journal marked as "modified"
%RMU-I-AIJISON, after-image journaling has been enabled
%RMU-W-DOFULLBCK, full database backup should be done to ensure future recovery
%RMU-I-LOGRECDB, recovering database file DEV:[DIR]TEST.RDB;1
%RMU-I-AIJAUTOREC, starting automatic after-image journal recovery
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-I-AIJONEDONE, AIJ file sequence 1 roll-forward operations completed
%RMU-W-NOTRANAPP, no transactions in this journal were applied
%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 2
$ RMU/DUMP/HEADER=BRIEF/OUTPUT=AFTER.TXT TEST
$
$ ! Compare the original database with the recovered database.
$ ! In this example, instead of the cache row length being changed
$ ! to 400 the number of database buffers is changed to 400.
$
$ DIFFERENCE BEFORE.TXT AFTER.TXT
.
.
.
************
File DEV:[DIR]BEFORE.TXT;1
   35         - Default user buffer count is 20
   36         - Default recovery buffer count is 20
   37         - Global buffers are disabled
******
File DEV:[DIR]AFTER.TXT;1
   35         - Default user buffer count is 400
   36         - Default recovery buffer count is 400 (stored as 20)
   37         - Global buffers are disabled
************
```

Depending on what row cache parameters were changed, various failures may occur in the RMU/RECOVER operation or in the database monitor. In the reported problem, RMU/RECOVER would fail with the following exception:

```
***** Exception at 007E35BC : PIO$FETCH + 000003EC
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000000000000, PC=00000000007E35BC, PS=0000001B
```

Also, the database monitor failed with the following exception:

```
**** Exception at hhhhhhhh : MON$DELETE_UNREFERENCED_GBL + 00000DAC
%SYSTEM-F-ACCVIO, access violation, virtual address=0000000000414000
```

To avoid this problem, do a full database and journal backup after altering any row cache parameters. If this problem is encountered, it is possible to recover the restored database up until the point in the journal that contains the row cache changes. That is, using the /UNTIL qualifier, recover the journals up to the point in time that the row cache changes were made.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.5.2 Recovered Database Corrupt When ROW SNAPSHOT IS ENABLED

When the snapshots in cache feature was enabled, it was possible for after−image journal (AIJ) entries to be logged with incorrect transaction sequence numbers (TSNs). This could result in a corrupt database if the journal was used to recover a restored database. The problem would occur if an error happened during an update statement and the transaction was later committed. For example, a constraint failure or lock timeout followed by a COMMIT could cause incorrect journal entries to be made.

This problem was introduced in Oracle Rdb Releases 7.1.4.4 and 7.2.0.2.

This problem can be avoided by setting all caches to ROW SNAPSHOT IS DISABLED.

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.6 RMU Show Statistics Errors Fixed

7.6.1 RMU/SHOW STATISTICS Hot Standby Statistics State Display Field

Bug 5396571

Previously, when using the TCP/IP network transport with the Hot Standby feature, the RMU /SHOW STATISTICS "Hot Standby Statistics" display "State:" field could overwrite the "UserSync:" heading as in the following example:

```
Node: HSVMS  (1/1/16)  Oracle Rdb V7.1-441 Perf. Monitor 18-JUL-2006 06:17:59.97
Rate: 3.00 Seconds           Hot Standby Statistics         Elapsed: 00:07:28.63
Page: 1 of 1  $1$DGA113:[MWILLEMS.HS.HS1.MASTER]MF_PERSONNEL.RDB;1  Mode: Online
-------------------------------------------------------------------------------
State: TCP/IP:72     rSync: Cold Current.Msg: 1       Cl Mstr.AIJ: 1:2
LagTime: 00:00:00 AutoSync: Cold Stalled.Msg: none     1 Stby.AIJ: 1:2
Stby.DB: HSVMS1::$1$DGA20:[MWILLEMS.HS.HS1.STANDBY]STANDBY_MF_PERSONNEL
```

The line starting with "State:" partly overwrites "UserSync:".

This problem has been corrected in Oracle Rdb Release 7.2.1.

7.6.2 RMU /SHOW STATISTICS Defined Logicals List Incomplete

Bug 5600122

Previously, it was likely that the RMU /SHOW STATISTICS Defined Logicals display did not properly list all logicals when the display was set to "Full" mode. This problem was caused by an incorrect calculation of the number of logical names possible.

This problem has been corrected in Oracle Rdb Release 7.2.1. The full list of logical names is correctly displayed.

7.6.3 Rdb Executive Sort and Temporary Work File Statistics

Bug 5617519

Previously, there was no reliable way to determine the number of sorting operations nor temporary work file operations within the Rdb executive at a database-wide level.

New statistics are available to help understand the number and type of sorting operations and temporary work file operations within the Rdb executive. These statistics are available on the "Rdb Executive Statistics" screen of the RMU /SHOW STATISTICS utility:

- records sorted − The number of data items passed in to a sort routine (note that the elements being sorted may not be actual database rows).
- quick−sorts − Sort operations that were handled entirely within an internally buffered quick sort routine. These are generally sorts of smaller cardinalities of simple sort keys.
- sort32 sorts − Sort operations that were handled by the internal SORT32 interface.
- workfile write IO − Write IO operations to SORT32 on−disk work files.
- workfile read IO − Read IO operations from SORT32 on−disk work files.
- temp file create − Creation of temporary relation work files.
- delete − Deletion of temporary relation work files.
- record put − Data written using RMS to temporary relation work files.
- record get − Data read using RMS from temporary relation work files.
- truncate − Rewind and truncate of temporary relation work files using RMS.
- record position − Rewind or backspace operation of temporary relation work files using RMS.

Example Rdb Executive Statistics display with sort and temporary work file statistics:

```
Node: LUCAS (1/1/1)    Oracle Rdb X7.2−00 Perf. Monitor  26−OCT−2006 22:41:29.41
Rate: 3.00 Seconds          Rdb Executive Statistics        Elapsed: 00:02:50.14
Page: 1 of 1           $1$DGA203:[HSEC]MF_PERSONNEL.RDB;1           Mode: Online
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
statistic.........      rate.per.second.............. total....... average......
name..............      max..... cur..... avg....... count....... per.trans....
queries compiled              16        0        0.7           82          16.4
index scans                   84        0        3.8          430          86.0
 index only                    0        0        0.0            0           0.0
 index full                   21        0        1.0          112          22.4
dynamic optimizer             10        0        0.3           34           6.8
 one abandoned                 0        0        0.0            1           0.2
 all abandoned                 0        0        0.0            0           0.0
records sorted                37        0        3.0          339          67.8
 quick−sorts                   0        0        0.0            3           0.6
 sort32 sorts                  0        0        0.0            4           0.8
  workfile write IO            0        0        0.0            0           0.0
  workfile read IO             0        0        0.0            0           0.0
temp file create              63        0        2.7          301          60.2
          delete              63        0        2.7          301          60.2
        truncate               0        0        0.0            0           0.0
      record put             246        0       11.3         1268         253.6
      record get             267        0       11.3         1268         253.6
 record position              63        0        2.7          301          60.2
```

Chapter 8
Enhancements And Changes Provided in Oracle Rdb Release 7.2.2.0

8.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.2.0

8.1.1 Reduced Executable Image Sizes, 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
- Reduction in executable image file size
- Enhancements to the optional run−time routine native compiler on I64

8.1.2 New SET FLAGS Keyword to Control Optimization Level

Bug 6389282

The optimization levels TOTAL TIME and FAST FIRST can be specified in the following ways:

- on the query itself using the OPTIMIZE FOR clause,
- within the query environment using the SET OPTIMIZATION LEVEL command,
- during application compile using SQL pre−compiler option /SQLOPTIONS=OPTIMIZATION_LEVEL, or the SQL Module Language /OPTIMIZATION_LEVEL qualifier,
- and specified via a query outline with the EXECUTION OPTIONS clause.

However, some dynamic SQL environments generate queries that cannot be affected by any of these methods. Therefore, Oracle Rdb has added a new flag to SET FLAGS (and RDMS$SET_FLAGS logical name) that can cover these types of queries.

The flag OPTIMIZATION_LEVEL can be used to change the default optimization level for a query. If the query explicitly uses the OPTIMIZE FOR clause or is compiled within an environment which overrides the default using the methods listed above, then no change will occur to the query optimization. If the query uses the default optimization level then their optimization will be modified by this flag.

- OPTIMIZATION_LEVEL with no option list (or an empty options list) will default to TOTAL TIME.
- NOOPTIMIZATION_LEVEL will revert to the default Oracle Rdb behavior.
- OPTIMIZATION_LEVEL(FAST_FIRST) will establish FAST FIRST as the default for queries in all sessions.
- OPTIMIZATION_LEVEL(TOTAL_TIME) will establish TOTAL TIME as the default for queries in all sessions.

- No other options are accepted for this keyword.

The following example shows the change of behavior for a query using the dynamic optimizer.

```
SQL> -- show with default behavior (FFirst tactic used)
SQL> select *
cont> from xtest
cont> where col2 between 999980 and 1000000
cont>   and col1 > 0
cont> ;
Tables:
  0 = XTEST
Leaf#01 FFirst 0:XTEST Card=10
  Bool: (0.COL2 >= 999980) AND (0.COL2 <= 1000000) AND (0.COL1 > 0)
  BgrNdx1 XTEST_IDX [1:0] Fan=17
    Keys: 0.COL1 > 0
0 rows selected
SQL>
SQL> -- use SET FLAGS
SQL> set flags 'optimization_level(total_time)';
SQL>
SQL> -- show that BgrOnly is used for TOTAL TIME
SQL> select *
cont> from xtest
cont> where col2 between 999980 and 1000000
cont>   and col1 > 0
cont> ;
Tables:
  0 = XTEST
Leaf#01 BgrOnly 0:XTEST Card=10
  Bool: (0.COL2 >= 999980) AND (0.COL2 <= 1000000) AND (0.COL1 > 0)
  BgrNdx1 XTEST_IDX [1:0] Fan=17
    Keys: 0.COL1 > 0
0 rows selected
SQL>
```

This feature has been added in Oracle Rdb Release 7.2.2.

8.1.3 New /ABMS_ONLY Qualifier to Only Dump Rdb Database ABM Pages

Currently, Oracle Rdb database Area Bit Map (ABM) pages can be dumped along with other types of pages for uniform storage areas or logical areas within uniform storage areas using the RMU/DUMP command. A new qualifier has been added to the RMU/DUMP command, /ABMS_ONLY, which will only dump ABM pages in uniform storage areas or in logical areas contained within uniform storage areas. The ABM pages can be dumped within a limited page range specified by the existing /START=n and/or /END=n qualifiers, where n is a page number; or if a limited page range is not specified, all ABM pages within uniform storage areas or within logical areas contained in uniform storage areas can be dumped.

The /ABMS_ONLY qualifier cannot be negated and will not be the default. The default if /ABMS_ONLY is not specified will continue to be to dump ABM pages along with other types of database pages for storage areas and logical areas. If the existing /AREAS or /LAREAS or /ALL_LIVE or /ALL_LAREA qualifiers are not used with the /ABMS_ONLY qualifier in the RMU/DUMP command line, a default of /ALL_LIVE is assumed to dump only ABM pages contained within all live uniform storage areas. The dump format for the ABM page has not changed. Headers will be output as they are currently to identify the live uniform storage

area being dumped and/or the logical area within the live uniform storage area followed by the ABM pages contained within each logical area. A dump of the database header will be included at the start of the dump in the same cases where it is currently included.

If there are no ABM pages within the specified page range or the storage area is a mixed format area or the logical area is contained within a mixed storage area, no ABM pages will be dumped since there are no ABM pages in these cases. If you execute the RMU/DUMP/HEADER command, the entries for storage areas defined for the Rdb database will specify which areas are of uniform format and which are of mixed format. The /ABMS_ONLY qualifier cannot be specified in the same dump command as the existing /SPAMS_ONLY qualifier which only dumps Space Management (SPAM) pages. The /ABMS_ONLY qualifier cannot be specified in the same dump command with the existing /SNAPSHOTS qualifier for dumping SNAPSHOT areas.

The syntax for dumping only ABM pages is as follows:

```
/ABMS_ONLY
```

In the following example, all ABM pages contained in all uniform storage areas in the specified Rdb database are dumped.

```
$ rmu/dump/abms_only/out=dmp.out mf_personnel
```

In the following example, only the ABM pages contained in the named uniform storage area in the specified Rdb database are dumped.

```
$ rmu/dump/abms_only/area=rdb$system mf_personnel
```

In the following example, only the ABM pages contained in the named logical area in a uniform storage area in the specified Rdb database are dumped.

```
$ rmu/dump/abms_only/larea=rdb$relations mf_personnel
```

In the following example, only the ABM pages contained within the specified page range in the named uniform storage area in the specified Rdb database are dumped.

```
rmu/dump/abms_only/area=rdb$system/start=1/end=5 mf_personnel
```

8.1.4 RMU/BACKUP Performance Enhancement

With storage areas located on different devices, an RMU/BACKUP would start reading all storage areas from the first input device before proceeding to the next input device. This caused an imbalance of I/O loads on the various devices. While one input device was highly active performing read I/Os, other input devices were idling.

As an enhancement in this release, RMU/BACKUP assigns storage areas to be saved to reader threads by using a round−robin scheme based on the disk devices of the storage areas. Larger storage areas for this disk device are selected first. As in the past, it still balances the amount of data that goes to each output device or save set or media−manager stream.

```
$ RMU/BACKUP/LOG=FULL LDA100:[DB]MYTESTDB −
    $1$DGA20:[BACKUP]RBF1,$1$DGA40:[BACKUP]RBF2/DISK=WRITERS=2
```

```
%RMU-I-BCKTXT_01, Writer thread 1 writes $1$DGA20:[BACKUP]RBF1.RBF; containing:
    File LDA100:[DB]MYTESTDB.RDA;1    1404 blocks
    File LDA100:[DB]A0007.RDA;1       5388 blocks
    File LDA300:[DB]A0006.RDA;1       7758 blocks
    File LDA100:[DB]A0004.RDA;1       4490 blocks
    File LDA200:[DB]A0005.RDA;1       9310 blocks
    File LDA300:[DB]A0009.RDA;1       4490 blocks
    File LDA200:[DB]A0011.RDA;1       3742 blocks
    File LDA100:[DB]A0001.RDA;1       2599 blocks
    File LDA300:[DB]A0012.RDA;1       3742 blocks
%RMU-I-BCKTXT_01, Writer thread 2 writes $1$DGA40:[BACKUP]RBF2.RBF; containing:
    File LDA200:[DB]A0002.RDA;1      27802 blocks
    File LDA100:[DB]A0010.RDA;1       3742 blocks
    File LDA300:[DB]A0003.RDA;1       4490 blocks
    File LDA200:[DB]A0008.RDA;1       2166 blocks
%RMU-I-BCKTXT_00, Backed up root file LDA100:[DB]MYTESTDB.RDB;1
%RMU-I-RESUME, resuming operation on volume 2 using _$1$DGA40
%RMU-I-BCKTXT_02, Starting full backup of storage area (RDB$SYSTEM)
LDA100:[DB]MYTESTDB.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (RDB$SYSTEM)
LDA100:[DB]MYTESTDB.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0009)
LDA300:[DB]A0009.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0002)
LDA200:[DB]A0002.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0007)
LDA100:[DB]A0007.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0010)
LDA100:[DB]A0010.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0006)
LDA300:[DB]A0006.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0003)
LDA300:[DB]A0003.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0004)
LDA100:[DB]A0004.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0008)
LDA200:[DB]A0008.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0005)
LDA200:[DB]A0005.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0008)
LDA200:[DB]A0008.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0010)
LDA100:[DB]A0010.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0003)
LDA300:[DB]A0003.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0004)
LDA100:[DB]A0004.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0011)
LDA200:[DB]A0011.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0009)
LDA300:[DB]A0009.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0001)
LDA100:[DB]A0001.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0007)
LDA100:[DB]A0007.RDA;1
%RMU-I-BCKTXT_02, Starting full backup of storage area (A0012)
LDA300:[DB]A0012.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0001)
LDA100:[DB]A0001.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0002)
LDA200:[DB]A0002.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0006)
```

```
LDA300:[DB]A0006.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0011)
LDA200:[DB]A0011.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0005)
LDA200:[DB]A0005.RDA;1
%RMU-I-BCKTXT_12, Completed full backup of storage area (A0012)
LDA300:[DB]A0012.RDA;1
%RMU-I-COMPLETED, BACKUP operation completed
```

8.1.5 /NOOUTPUT Can Now Be Specified With the RMU/SET SERVER Command

The "RMU /SET SERVER /OUTPUT=filespec servertype" command can be used to specify the default output file specification for several of the database server processes. If the output file specification is empty, the output file entry is disabled. Now, in addition to specifying an empty output file specification with the /OUTPUT qualifier in order to disable the output file entry, "RMU /SET SERVER /NOOUTPUT servertype" can be specified as a way to disable the output file entry. Note that /NOOUTPUT is the default and if /OUTPUT is not specified the output file server logging entry will be disabled.

The syntax for the /OUTPUT qualifier is therefore the following.

```
[NO]OUTPUT[=filespec]
```

The following example shows that now the output file entry can be disabled by the RMU/SET SERVER command by specifing /NOOUTPUT or by specifying /OUTPUT with an empty output file specification or by not specifying /OUTPUT since /NOOUTPUT is the default.

```
$ RMU /SET SERVER LRS /NOOUTPUT DUA0:[ZDB]ZDB.RDB
$ RMU /SET SERVER LRS /OUTPUT="" DUA0:[ZDB]ZDB.RDB
$ RMU /SET SERVER LRS DUA0:[ZDB]ZDB.RDB
```

8.1.6 New Configuration Parameter SQL_PROTOCOL_RETRY

Bug 6494061

In prior versions of Oracle Rdb, the client side of a remote connection would, by default, try to use an older version of remote Rdb protocol to communicate with the remote server if there were an error using the current version. This retry would occur for a failure on database attach, create, or delete. The old protocol version was the one used by Rdb 5.1 and earlier. The intent of this retry was to allow the client to communicate with very old versions of the remote server automatically.

The affect of the retry for most users was that it took twice as long and twice as many resources for a failure on an attach, create, or delete database to be reported. In some cases, it was also possible for the client and server protocol exchange to become so confused that the client would hang in a LEF state. This sometimes happened, for example, if the Rdb monitor were not running on the node where the remote server was executing.

In order to address the issues described above, the default behavior has been changed so that there is no retry using the old protocol. For users needing to connect to a remote database using a Rdb 5.1 or earlier version of the remote server, a new configuration file parameter has been added to cause Rdb to retry using the old protocol. This configuration file parameter is named SQL_PROTOCOL_RETRY and should be placed in the RDB$CLIENT_DEFAULTS.DAT file. See Chapter 4 of the Oracle Rdb Installation and Configuration Guide for information on configuration files.

For example, a client application using a remote database will retry using the old remote protocol if the RDB$CLIENT_DEFAULTS.DAT contains the following entry:

```
SQL_PROTOCOL_RETRY TRUE
```

8.1.7 RMU /RESTORE Allows Change of Page Size For Uniform Format Storage Areas

Bug 705542

Previously, the RMU/RESTORE command allowed increasing the page size only for mixed−format storage areas.

This restriction has been relaxed. Page size may now be increased for both mixed and uniform storage area formats during an RMU restore operation.

Chapter 9
Enhancements And Changes Provided in Oracle Rdb Release 7.2.1.4

9.1 Enhancements And Changes Provided in Oracle Rdb Release 7.2.1.4

9.1.1 RMU/SHOW STATISTICS /STALL_LOG And /ALARM

Enhancement Bug 6331702

In previous Oracle Rdb releases, using the RMU/SHOW STATISTICS /STALL_LOG feature resulted in every active stall message being written to the stall log file at each sample interval. In some cases, this is an excessive amount of information as a result of most of the active stalls being of short duration.

This problem has been corrected in Oracle Rdb Release 7.2.1.4. The RMU/SHOW STATISTICS command now accepts a new LOG_STALL_ALARM keyword for the /OPTION qualifier. If LOG_STALL_ALARM is present when using /STALL_LOG and /ALARM, only those stalls exceeding the /ALARM specified duration are written to the stall log output file.

9.1.2 RMU/SHOW STATISTICS Stall Alarm Invoked Procedure Parameter Addition

Enhancement Bug 6331702

Parameter P6 has been added to the parameters passed to the stall alarm invoked command. The P6 parameter contains the formatting stall message string.

The parameters passed to the invoked command are the following:

Table 9−1 Stall Alarm Invoked Procedure Parameters

Parameter	Description
P1	Root file specification
P2	Current data/time
P3	Process ID
P4	Stream ID
P5	Alarm threshold seconds
P6	Formatted stall message

9.1.3 RMU /SHOW AIP New Qualifier /BRIEF

Enhancement Bug 3390639

The RMU /SHOW AIP command now supports the qualifier /BRIEF to display AIP information in a condensed, tabular form as in the following example:

```
$ RMU/SHOW AIP  DKA0:[DB]DB /PAREA=(4,5)/BRIEF
*-------------------------------------------------------------------------------
* Logical Area Name             LArea PArea   Len Type
*-------------------------------------------------------------------------------
RDB$SYSTEM_RECORD                  60     4   215 SYSTEM RECORD
RDB$SYSTEM_RECORD                  61     5   215 SYSTEM RECORD
EMPLOYEES_HASH                     79     4   215 HASH INDEX
EMPLOYEES                          82     4   121 TABLE
JOB_HISTORY_HASH                   85     4   215 HASH INDEX
JOB_HISTORY                        88     4    42 TABLE
DEPARTMENTS_INDEX                  89     5   430 SORTED INDEX
DEPARTMENTS                        90     5    55 TABLE
```

The columns displayed include:

- Logical Area Name − Name of the logical area stored in the AIP entry
- LArea − Logical area number stored in the AIP entry
- PArea − Physical area number stored in the AIP entry
- Len − Object length stored in the AIP entry
- Type − Object type stored in the AIP entry. The following object types may be displayed:
  - UNKNOWN − The logical area type is unknown or has not been set
  - TABLE − A data table type
  - SORTED INDEX − A sorted index type
  - HASH INDEX − A hashed index type
  - SYSTEM RECORD − A system record type
  - LARGE OBJECT − A large object (BLOB) type

9.1.4 RMU /SHOW AIP New Qualifier /PAREA

The RMU /SHOW AIP command now supports the qualifier /PAREA to display AIP information for logical areas stored in the specified physical areas.

```
$ RMU/SHOW AIP DKA0:[DB]DB /BRIEF /PAREA=(4,5)
*-------------------------------------------------------------------------------
* Logical Area Name             LArea PArea   Len Type
*-------------------------------------------------------------------------------
RDB$SYSTEM_RECORD                  60     4   215 SYSTEM RECORD
RDB$SYSTEM_RECORD                  61     5   215 SYSTEM RECORD
EMPLOYEES_HASH                     79     4   215 HASH INDEX
EMPLOYEES                          82     4   121 TABLE
JOB_HISTORY_HASH                   85     4   215 HASH INDEX
JOB_HISTORY                        88     4    42 TABLE
DEPARTMENTS_INDEX                  89     5   430 SORTED INDEX
DEPARTMENTS                        90     5    55 TABLE
```

9.1.5 New Options for RMU DUMP EXPORT Command

This release of Oracle Rdb adds new keywords to the OPTIONS qualifier for RMU Dump Export. The OPTIONS qualifier allows the user to modify the output from this dump command.

- ALLOCATION
  When importing databases for testing, the full allocation recorded in the interchange file is often not

required. The clauses ALLOCATION and SNAPSHOT ALLOCATION are controlled by this option. The default is ALLOCATION. Use NOALLOCATION to omit these clauses from the generated SQL script. This option is ignored if NOIMPORT_DATABASE is specified or defaulted for the OPTIONS qualifier.

- FILENAME_ONLY
  When importing databases for testing, the full file specification for the database root, storage areas and snapshot areas recorded in the interchange file is often not required. The FILENAME clauses are controlled by this option which trims the specification to only the filename portion. The default is NOFILENAME_ONLY. Use FILENAME_ONLY to truncate the file specification in generated SQL script. This option is ignored if NOIMPORT_DATABASE is specified or defaulted for the OPTIONS qualifier.
- HEADER_SECTION
  This option allows the database administrator to display just the header portion of the interchange file and avoid dumping the data or metadata for every row in the table.

```
$ RMU/DUMP/EXPORT/OPTION=HEADER JOBS.UNL

BEGIN HEADER SECTION - (0)
    NONCORE_TEXT HDR_BRP_ID - (20) :  Load/Unload utility
    CORE_NUMERIC HDR_BRPFILE_VERSION - (1) : 4
    NONCORE_TEXT HDR_DBS_ID - (18) : Oracle Rdb V7.2-10
    NONCORE_TEXT HDR_DB_NAME - (16) : DB$:MF_PERSONNEL
    NONCORE_DATE HDR_DB_LOG_BACKUP_DATE - (8) :  3-JUL-2006 16:52:32.83
    CORE_NUMERIC HDR_DATA_COMPRESSION - (1) : 1
END HEADER SECTION - (0)

$
```

  In this example, the output describes the creator of the interchange file (RMU/UNLOAD), the version of Rdb used to create the file, the file specification of the database used, the date and time the interchange file was created, and an indication that compression was used by RMU Unload.
- IMPORT_DATABASE
  This keyword requests that the output from RMU Dump Export be formatted as a SQL IMPORT DATABASE statement. It uses the database attributes present in the interchange file formatted as SQL clauses. Of particular interest is the CREATE STORAGE AREA clauses which are required to IMPORT the source interchange (.rbr) file.
  The keyword HEADER_SECTION is implicitly selected when IMPORT_DATABASE is used, limiting the I/O to the interchange file to the section containing the database attributes.
  The default is NOIMPORT_DATABASE.

Usage Notes

- If the source interchange file is created by RMU Unload, then it does not contain any IMPORT DATABASE information and the generated SQL script cannot be used to create a database from such an interchange file.

```
$ RMU/DUMP/EXPORT/OP=IMPORT_DATABASE EMPLOYEES.UNL/OUT=EMP.SQL
$ SQL$ @EMP.SQL
SQL> IMPORT DATABASE
cont>     from 'DISK1:[TESTING]EMPLOYEES.UNL;1'
cont>     -- ident ' Load/Unload utility'
cont>     -- backup file version 4
cont>     -- database ident 'Oracle Rdb V7.2-131'
```

```
cont>     filename 'DB$:MF_PERSONNEL'
cont> ;
%SQL-F-EXTRADATA, unexpected data at the end of the .RBR file
$
```

- The IMPORT_DATABASE option is intended to create a SQL script as an aid to the database administrator. Some editing of the generated script may be required under some circumstances. Only a subset of the database attributes are dumped by RMU for the IMPORT_DATABASE output. Continue to use the RMU Dump Export Option=NOIMPORT_DATABASE to see all attributes recorded in the interchange file.

Chapter 10
Documentation Corrections, Additions and Changes

This chapter provides corrections for documentation errors and omissions.

10.1 Documentation Corrections

10.1.1 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>
```

10.1.2 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.

10.1.3 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.

10.1.4 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;
```

10.1.5 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.

10.1.6 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 10−1.

Table 10−1 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.

10.1.7 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.

10.1.8 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.

10.1.9 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.

10.2 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
```

10.3 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 11
Known Problems and Restrictions

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

11.1 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.

11.1.1 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.

11.1.2 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.

11.1.3 Response 'QUIT' to RMU Restart Prompt Loops

Bug 6001187

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

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

11.1.4 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

11.1.5 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)

11.1.6 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.

11.1.7 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.

11.1.8 SQL Procedure External Location Should Be Upper Case

Bug 4722422

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

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

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

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

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

11.1.9 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.

11.1.10 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
```

11.1.11 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.

11.1.12 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.

11.1.13 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.

11.1.14 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.

11.1.15 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.

11.1.16 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.

11.1.17 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.

11.1.18 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.

11.1.19 Both Application and Oracle Rdb Using SYS$HIBER

In application processes that use Oracle Rdb and the $HIBER system service (possibly through RTL routines such as LIB$WAIT or within multi−threaded processes (including JAVA−based) using PTHREAD$RTL), the application must ensure that the event being waited for has actually occurred. Oracle Rdb uses $HIBER/$WAKE sequences for interprocess communications.

The use of the $WAKE system service by Oracle Rdb can interfere with other users of $HIBER (such as the RTL routine LIB$WAIT) that do not check for event completion, possibly causing a $HIBER to be unexpectedly resumed without waiting at all. In other cases, it can be possible for one caller of the SYS$HIBER service to "consume" a wakeup or a pending wakeup that might have been intended for some other SYS$WAKE. This is even more likely when using Rdb in that Rdb's use of SYS$HIBER/SYS$WAKE can occur at an inner access mode.

To avoid these situations, consider altering the application to use a code sequence that avoids continuing without a check for the operation (such as a delay or a timer firing) being complete. All uses of hiber/wake within the application must be coded correctly to deal with spurrious wakes at any time.

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

```
OWN WAKEFLG : VOLATILE; ! Volatile to force memory fetch

ROUTINE TIMER_WAIT:
    BEGIN
    WAKEFLG = FALSE ! Clear timer flag

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

ROUTINE TIMER_AST:
    BEGIN
    WAKEFLG = TRUE ! Set flag indicating timer expired
    STAT = SYS$WAKE () ! Wake the main−line code
    IF STAT <> SS$_NORMAL THEN LIB$SIGNAL (STAT)
    END
```

The LIB$K_NOWAKE flag can be specified when using the OpenVMS LIB$WAIT routine to allow an alternate wait scheme (using the $SYNCH system service) that can avoid potential problems with multiple code sequences using the $HIBER system service.

In order to prevent application hangs, inner−mode users of SYS$HIBER must take additional steps to ensure that a pending wake is not errantly " consumed ". The general way of accomplishing this is to issue a SYS$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.

11.1.20 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.

11.1.21 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.

11.1.22 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 11−1 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.

11.1.23 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.

11.2 SQL Known Problems and Restrictions

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

11.2.1 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.

11.2.2 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.

11.2.3 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.

11.2.4 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.

11.2.5 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.

11.3 Oracle RMU Known Problems and Restrictions

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

11.3.1 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
```

11.3.2 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.

11.3.3 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.

11.3.4 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.

11.3.5 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.

11.3.6 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.

11.4 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.

11.4.1 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.

11.4.2 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.

11.4.3 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.

11.4.4 Strict Partitioning May Scan Extra Partitions

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

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

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

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

11.4.5 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.

11.4.6 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.

11.4.7 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.

11.5 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.

11.5.1 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.)

11.5.2 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.

11.5.3 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 11−2 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

11.5.4 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.

11.5.5 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.

11.5.6 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